uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Dong Liu

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A QoS Capable Architecture for IEEE 802.11 Wireless LAN and Prototype Implementation on Linux

TITRE DE LA THÈSE / TITLE OF THESIS

D. Makrakis

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

A. Miri

J. Talim

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

A QoS Capable Architecture for IEEE 802.11 Wireless LAN and the Prototype Implementation on Linux

By
Dong Liu

A thesis submitted to
the Faculty of Graduate Studies
University of Ottawa
in partial fulfillment of the requirements
for the degree of

Master of Applied Science
in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical Engineering
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa
Ottawa, Ontario, Canada

April 2008

Your file Votre référence
ISBN: 978-0-494-50902-9
Our file Notre référence
ISBN: 978-0-494-50902-9

ABSTRACT

The increasing popularity of wireless handheld devices with multimedia functionality and IEEE 802.11 based network connectivity generated the need for Quality of Service (QoS) support of real time applications deployed over the wireless LANs. However, the time varying nature of wireless link and network is a serious challenge when attempting to provide Quality of Service support.

This thesis proposes a low cost/complexity QoS capable wireless LAN architecture. We designed a distributed network congestion control scheme based on current commercial available hardware. The proposed technique was implemented on a Linux based testbed. Its performance was assessed through extensive experiments.

Our evaluation test work includes two parts. First an IEEE802.11b wireless LAN testbed is established and a thorough network capacity measurement test is performed. The actual performance of real wireless LAN products is evaluated in a typical indoor environment. Furthermore, the impacts of multiple competing stations and nearby radio noise are also evaluated in this testbed.

In the second part, the QoS capable wireless architecture prototype is implemented using Linux operating system Redhat 7.3 and Orinoco Golden IEEE802.11b wireless network card. The performance evaluation is conducted using the prototype implemented on the IEEE802.11b wireless LAN testbed.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere appreciation to my supervisor, Dr. Dimitrios Makrakis, for his inspirational and invaluable support throughout this research work. Dr. Makrakis shared the wealth of his knowledge and experience with me and always gave me creative ideas and constructive comments.

All my colleagues in our research group have contributed to establishing a productive work environment. I also would like to thank Mr. Quanyou Zhou, Bin Zhou and Mrs. Helen Zhang for their valuable advices and friendly help.

I would like to give my loving thanks to my wife, Wei Zhang and son, David Liu, for their encouragement and understanding. Without them, I would have never finished this work. I also like to give special thanks to my parents and parents-in-law for always believing in me and helping me.

LIST OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONUMS

AF	Assured Forward
AIFS	Arbitration Inter-frame Space
AIMD	Additive Increase Multiplicative Decrease
AP	Access point
BE	Best Effort
BSS	Basic Service Set
CCK	Complementary Code Keying
CRC	Cyclic Redundancy Code
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear To Send
CP	Competition Period
CFP	Contention Free Period
DCF	Distributed Coordination Function
DBPSK	Differential Binary Phase Shift Keying
DiffServ	Differentiated Services
DIFS	DCF Interframe space
DS	Distributed system
DQPSK	Differential Quadrature Phase Shift Keying
DSCP	Differentiated Services Code Point
DSSS	Direct Sequence Spread Spectrum
DWFQ	Distributed Weighted Fair Queuing
EDCF	Enhanced version of DCF

EF	Expedited Forwarding
ESS	Extended service set
ESSID	Extended service set identification
FA	Foreign Agent
FCS	Frame Checks Sequence
FEC	Forwarding Equivalent Class
HCF	Hybrid Coordination Function
F-FSC	Hierarchical Fair Service Curve Algorithm
FHSS	Frequency-Hopping Spread Spectrum
GSM	Global System for Mobile communication
IBSS	Independent Basic Service Set
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IFS	Interframe Space
InterServ	Integrated Services
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
LAN	Local Area Network
NAV	Network Allocation Vector
NIC	Network Interface Card
MAC	Medium Access Control
MSDU	Medium Access Control Service Data Unit

OFDM	Orthogonal Frequency Division Multiplexing
PBCC	Packet Binary Convolutional Coding
PCF	Point Coordination Function
PHB	Per-Hop Behavior
PIFS	PCF Interframe Space
PLCP	Physical Layer Convergence Procedure
QoS	Quality of Service
RTS	Request To Send
RFC	Request for Comments
RSVP	Resource Reservation Protocol
SIFS	Short Interframe Space
SCFQ	Self-Clocked Fair Queuing
SWAN	Stateless Wireless Ad Hoc Networks
TXOP	Transmission Opportunity
TCA	Traffic Categories Aggregate
TOS	Type of Service
TTL	Time-To-Live
VoIP	Voice over IP
WFQ	Weighted Fair Queuing
WLAN	Wireless LAN

Chapter 1

Introduction

In recent years, the telecommunications technology has evolved at a fast pace. People are able to access the Internet from different locations using various kinds of end devices enabled with wireless connectivity. As we move in a fast pace from the era of networking to the era of pervasive computing, it is expected that over the coming year the number of wireless devices such as cellular phones, personal digital assistants (PDA) will increase dramatically.

Among the various wireless technologies, the IEEE 802.11 wireless LAN (WIFI) [WLAN], [WLANB], [WLANA], [PRA01] was originally designed to provide users with local, low cost wireless data access. In recent years, more and more telecommunication service providers have turned to the IEEE802.11 wireless LAN for higher wireless data transmission speed. "Hot spot" wireless LANs have been established in airports, malls, cafe shops, trains and other public areas, providing the nomadic user with fast wireless Internet access.

The unreliable nature of the wireless transmission media makes packet scheduling in the wireless LAN environment a challenging problem. Recent research work has made considerable progress, producing algorithms that are able to provide fair sharing of a wireless link [CAO01], [FAT02], [LU97], [LU99], [NAN99], [ABH93], [DEM89], [FLO95], [SHR95], [STO97], [KES91], [GOY96]. Moreover, several new proposals for the implementation of QoS capable technologies are in progress, such as the IEEE802.11e [MANG02].

1.1 Previous Work

Considerable work has been performed in the past to determine the actual capacity of the IEEE 802.11 wireless LAN [BIA00A], [BIA00B], [CAL98], [CHH97], [CRO97]. The authors of [XYL99] conducted comprehensive measurements using a commercial WLAN of 2 Mbps transmission speed for the purpose of assessing the performance of UDP/TCP protocols over IEEE 802.11 networks. In [ECK96] and [HAD02], the authors evaluated the performance of the wireless LAN in the presence of Rayleigh fading, the near/far effect and the error characteristics occurring in indoor wireless networks.

In order to incorporate QoS support to the IP network, the Internet Engineering Task Force (IETF) standardized an extension to the Internet architecture and protocols: the Integrated Services (InterServ) [BRA94]. The Resource Reservation Protocol (RSVP) [BRA97] is used to provide a way to specify and provide Quality of Service within an IP network by reserving network resource along the path through which the traffic is transmitted. However, it is difficult to deploy Interserv in large networks, because Interserv is not scalable due to the fact that it needs to "administer" the network resources on a per flow basis. Another architecture, Differentiated Services (DiffServ) [BLA98], was developed and standardized in order to overcome this shortcoming of Interserv. It does so by aggregating several flows into one common per hop behavior (PHB).

An important concept in QoS support is the packet scheduling discipline. Considerable research has been performed on scheduling algorithms [CAO01] [FAT02]. Scheduling algorithms provide the mechanism for bandwidth allocation and multiplexing at the packet level. Many scheduling algorithms capable of providing certain QoS guarantee have been developed for wired networks, such as Hierarchical Fair Service Curve Algorithm (HFSC), Link-sharing and Resource Management Models, etc. [ABH93], [DEM89], [FLO95], [SHR95], [STO97], [KES91], [GOY96]. However these scheduling algorithms cannot be directly deployed in wireless networks, because the wireless communication networks have some special characteristics not existing in wired networks. These issues include: the poor quality of the wireless communication channels which experience high error rate; the location-dependent and time-varying link capacity; the lower capacity of the wireless

network when compared to the capacity available at wired networks of the similar complexity; the user mobility creates serious challenge to service provision. A packet scheduling technique proposed for wireless networks is the Idealized Wireless Fair Queuing (IWFQ) [LU97], [LU99], [NAN99]. IWFQ realizes the wireless fluid fair queuing model based on two key assumptions: (a) each flow knows whether it can transmit correctly in the current slot and (b) packets can be tagged as soon as they arrive. The Channel Condition Independent Packet Fair Queuing (CIF-Q) [EUG98] scheduling scheme is a modification of the Start-time Fair Queuing (SFQ) [GOV96] that fits better to the wireless channel. There are three flow states, indicating the real service a flow receives compared to the service it should have obtained in the error-free reference system.

Recently, several QoS enhancement mechanisms have been proposed for the IEEE802.11 wireless LAN. The Distributed Weighted Fair Queuing (DWFQ)[BAN02] is a distributed QoS scheme, which extends the distributed coordination function (DCF) mode of the 802.11 MAC protocol to become capable of distributing the bandwidth of the wireless LAN among the different traffic flows proportionally to their weights. The Distributed Fair Scheduling (DFS) [VAI00] emulates the Self-clocked Fair Queuing (SCFQ) [GOL94] in a distributed manner, by modifying the DCF of IEEE802.11 wireless LAN MAC protocol. The Distributed Deficit Round Robin (DDRR) [PAT02] is proposed to emulate the Deficit Round Robin (DRR) [SHR95] at the MAC sub-layer of the IEEE802.11 wireless LAN. Some other interesting works related to packet scheduling in wireless networks could be found in [ECK00], [FANG02], [RAN99], [MOO00].

However, although the majority of the currently available wireless scheduling algorithms can provide an improved link-sharing ability, they need to have accurate knowledge of the quality of the wireless channel at the moment the packet is transmitted. Also these scheduling schemes are designed to be implemented at the link layer, which requires the support of the various vendors who produce the wireless card. Without changing of hardware, these techniques are unable to be deployed, meaning that use of currently available, inexpensive, off-the-shelf wireless cards is not possible. Wischhof and Lockwood [WIS01] proposed a method that uses the long-term channel-state information in order to

control the sharing of wireless bandwidth. The algorithm enables the operator of a WLAN to equalize the revenue/cost for each customer in the network and to control the link sharing based on a combination of user-centric and operator-centric factors. The simulation and prototype evaluation results showed that the modified HFSC algorithm is able to improve the controlled sharing of the wireless link without requiring modifications at the MAC layer.

In [BRE00], [CEN00], [BIA00B], endpoint admission control schemes that use end-to-end measurement-based admission control are proposed and analyzed for wired networks. Gahng-Seop [AHN02A], [AHN02B] proposed the Stateless Wireless Ad Hoc Network (SWAN), a stateless network model that uses distributed control algorithms to deliver service differentiation in mobile wireless ad hoc networks. The proposed architecture is designed to handle both, real-time UDP traffic, and best effort UDP and TCP traffic, without requiring the use and management of per-flow state information. The local rate control is used for best effort traffic and the sender-based admission control is for the real-time UDP traffic. SWAN uses Explicit Congestion Notification (ECN) to dynamically regulate admitted real-time sessions. However, the local rate control of SWAN is based on the information of the MAC layer Delay, which requires the support from the firmware of the wireless NICs. This hardware dependency means that the SWAN cannot be deployed using current off-the-shelf wireless NICs.

1.2 Objective and Contributions

In this thesis, a distributed network congestion control scheme is proposed and implemented on RedHat Linux operation system. This QoS scheme utilizes the time elapsing between the moment a packet is accepted by the driver of the WLAN Network Interface Card (NIC) and the moment the driver issues the next interrupt, indicating the readiness to accept the next packet, in order to determine the network congestion status. This parameter is termed as "NIC delay". Based on the NIC delay information, the distributed traffic rate controllers implemented in our scheme use the measured value of packet delay to decide the transmission rate and call flow policy that should be applied in order to avoid congestion.
We point out that the proposed scheme supports QoS service effectively, without requiring any forms of cross layer signaling between link and IP layers. No change of the functionality

of the standardized Distributed Coordination Function (DCF), operating on best-effort, is required, as well in order to support the quality of service. All software application and patches are performed within the computing device. The firmware of the NIC remains unchanged. This implication is that the proposed scheme can be implemented using existing commercially available WLAN NICs, leading to low-cost practical implementations. It only requires a "patch" to the standard function of Linux driver, instead of the link layer's software. Changing link layer's software is a tedious task, since it involves access to the firmware that is different from one vendor to the other and is deliberatively not easily programmable by users. We note that while the scheme and developed software were implemented and applied on IEEE802.11b system (the dominated WLAN standard at the time when the research was initiated), it can be applied to any more recent existing (e.g. IEEE 802.11(a)/(g)) and upcoming (IEEE 802.11h) standards, as the scheme and the software design are genetic and applied at the IP layer, of cause some adjustment of the system parameters might be required, in order to have the system offer the most possible when used with the alternative versions of the IEEE 802.11.

Before closing this section, we wish to point out that, similarity to the design of the new scheme, the task of software development, testbed implementation, running of tests, processing the measurement and interpretation of the results were tedious as well. The readers can take a good appreciation of the software development work through the Appendix A, where we clearly indicate the change and addiction made. Please note that prior to initiating modification, we have to thoroughly study and understand the software; a tedious, time-consuming task.

Chapter 6 and chapter 7 provide the achievement of the testbed. Reading through the chapters, the reader does not get an appreciation of the work involved in stabling, trouble shooting and testing the limitation of each node (e.g. how much traffic the node had enough power to perform reliably).

Coming to the collection of data, we have to run nine test cases. In order to reduce the possibility of having other LANs present while experiments were in the risk of having the

collected data be corrupted by interference, we made a habit to be running experiment late at night and always check before and during the experiment for presence of other LANs. In every test case, the results are collected for multiple runs with different testing parameters. Also, experiment was repeated for five times and in every run, 100,000 packets has been sent and captured for analysis.

The performance of the proposed scheme was assessed through extensive tests running on the RedHat Linux based test bed. The lab works for this thesis includes the following phases:

a. Set up the wireless LAN test bed

During this period, an IEEE802.11b wireless LAN test bed is setup. The testbed includes a access point, which is able to be a part of a mesh wireless network. The access point provides the footprint to users, which can be PCs, laptops and PDAs, as well as pass traffic to other access points. The setting up work includes:

Install Orinoco IEEE802.11b wireless card on RedHat Linux PC.

Install Lycsys IEEE802.11b wireless card on RedHat Linux.

Install and configure HostAP software, which enable the Linux PC working as an IEEE802.11b access point.

Write several test tool programs, including: PCAP based network traffic monitor; Raw socket based network traffic generator, which is able to mark the headers of IP packet in order to provide different service types and packet sequence number. Wireless channel quality monitor, etc.

b. Implemented the QoS scheme in RedHat Linux, kennel version 2.4.17

The function of the proposed QoS scheme is implemented using standard C program and is patched into the RedHat Linux kernel source tree, version 2.4.17. The main challenge for this task is to avoid breaking the functionality and stability of the running Linux kernel, while providing expected QoS functions. Implementing process includes:

Completely read and understand the Linux network stack, which realizes the TCP/IP protocol and includes hundred of source files.

Understand Linux Traffic Control module and HTB queuing discipline. Add a data rate aching callback function to HTB queuing discipline. This callback function enables our data rate controller to dynamically adjust the HTB data rate though in-kernel function callback, instead of using tc scripts, which do not need to change HTB function code but has a longer delay in data rate changing procedure.

Add NIC delay monitoring functionality to the Orinoco wireless card. The driver code (Orinoco.c) has been patched and recompiled into the Linux kernel, version 2.4.17.

c. IEEE802.11b network basic performance test

Tests are carried out to determine the limitations of the IEEE 802.11b wireless LAN network. We measured the real maximum capacity of the IEEE 802.11b wireless LAN in different data rate configuration and wireless channel conditions. We also collected the available information, such as wireless channel quality and NIC delay, from the driver of the wireless card used in this test.

By analyzing the relationship between the performance of the traffic streams and the collected information, we are able to understand the behavior of the IEEE 802.11b MAC layer and find/address the problems that may be hidden in the real network, which they should become identified before "building" the new system.

d. QoS Scheme performance test

Four performance evaluation test cases have been applied to the proposed QoS scheme, using on the IEEE802.11b wireless LAN testbed. The test results show that the bandwidth requirement of the EF traffic is satisfied, while the bandwidth of the BE traffic is as large as possible when the network resource is available. We also evaluated the effects of the system parameters, including decreasing factor, increasing factor and data rate-updating interval, on the performance of the QoS scheme.

1.3 Thesis Organization

The remaining of the thesis is organized as follows. In Chapter 2, the IEEE802.11 Wireless LAN standard is discussed in detail. In Chapter 3, we introduce two important QoS protocols, Integrated Service and Differentiated Service. The QoS enhancement technologies and algorithms developed for IEEE802.11 Wireless LAN are also introduced in Chapter 3. We describe our proposed QoS capable wireless LAN architecture in Chapter 4. In Chapter 5 we present the implementation of our QoS architecture. Chapter 6 provides the validation of our wireless testbed. In Chapter 7, the evaluation of the proposed QoS scheme is presented. We conclude our work in Chapter 8.

Chapter 2

Description of the IEEE802.11 Wireless LAN

In this chapter, the IEEE 802.11 wireless LAN standard [WLAN], [WLANB], [WLANA] is described. IEEE 802.11 is the most popular WLAN at present. It belongs to the group of 802.x LAN standards, to which the 802.3 (Ethernet) belong as well. The standard specifies the physical and medium access layers, which are designed to address the special requirements of the wireless network, but offers the same interface as other 802.x standards (Ethernet, token ring, etc.) to the network layer, in order to maintain interoperability. In Section 2.1, the system architecture of the wireless LAN is described. The physical layer of IEEE 802.11 is briefly introduced in Section 2.2. We focus on the MAC sublayer of IEEE 802.11 in Section 2.3.

2.1 System Architecture

The 802.11 wireless LAN standard suggests the supports the following two network topologies:

- Independent Basic Service Set (IBSS) networks.
- Extended Service Set (ESS) networks.

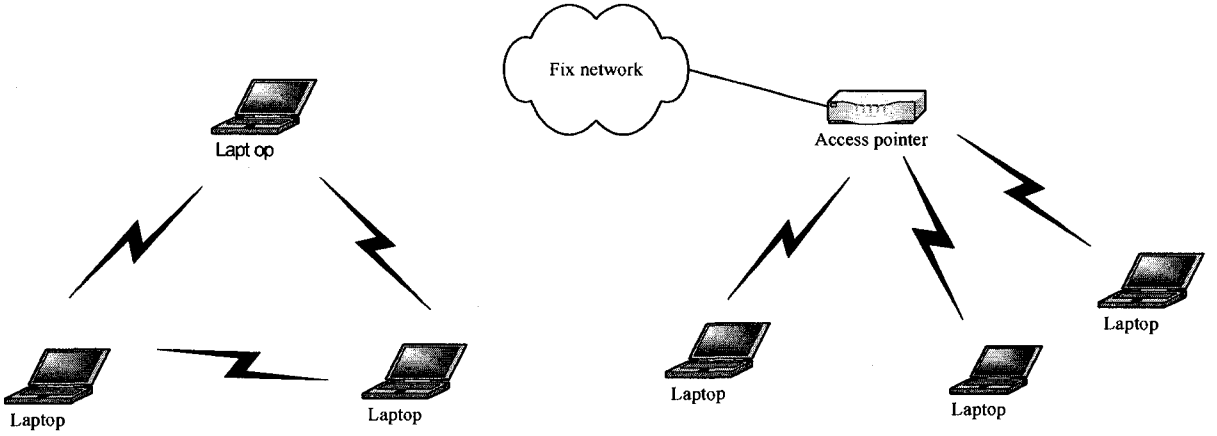
Both of the above networks utilize a basic block defined as Basic Service Set (BSS). A BSS provides a coverage in which stations of the BSS remain fully connected and are free to move within the BSS. However, the station cannot communicate directly with other stations if it leaves the BSS. Thus a BSS can be defined as a group of stations that are located in the same geographic area and are controlled by a single coordination function (distributed coordination function or point coordination function).

2.1.1 Independent Basic Service Set

Figure 2.1(a) displays a group of stations that belong to a single BSS. They are able to communicate with each other without the aid of any infrastructure network. This system is called Independent Basic Service Set (IBSS) or Ad Hoc network. There is no fixed structure in the Ad Hoc network. The stations that belong to the same Ad Hoc network specify the same Extended Service Set Identification (ESSID) and use the same radio frequency.

2.1.2 Extended Service Set

In figure 2.1(b), the Extended Service Set (ESS) forms an infrastructure network. This architecture uses the fixed access point (AP) through which mobile stations can communicate with each other or the nodes outside the BSS. The AP is a station that can provide Distributed System (DS) services to other stations. The DS services defined in IEEE 802.11 include: Association, Disassociation, Distribution, Integration and Reassociation. The ESS consists of multiple BSSs that are integrated together using a common distribution system, which is a backbone network that is responsible for the transport of Medium Access Control Service Data Units (MSDUs). This structure is very similar to the modern cellular networks that have been deployed around the world.



(a) AD HOC network

(b) Infrastructure network

Figure 2.1 Wireless LAN network structure

2.2 IEEE 802.11 Physical Layers

2.2.1 IEEE 802.11

IEEE 802.11 [WLAN] specifies three different physical layers: Frequency-Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS), and Infrared Physical layer (IR). IEEE 802.11a and IEEE 802.11g use another physical layer: Orthogonal Frequency Division Multiplexing (OFDM). The physical layer consists of the following two protocol functions [WLAN]:

a) A physical layer convergence function, which adapts the capabilities of the Physical Medium Dependent (PMD) system to the PHY service. This function is supported by the Physical Layer Convergence Procedure (PLCP), which defines a method of mapping the IEEE 802.11 MAC Sublayer Protocol Data Units (MPDUs) into a framing format suitable for sending and receiving user data and management information between two or more stations using the associated PMD system.

b) A PMD system, whose function defines the characteristics of, and the method of transmitting and receiving data through a Wireless Medium (WM) between two or more stations.

Direct sequence spread spectrum for the 2.4 GHz ISM band

The DSSS transmitting system in IEEE 802.11 provides a wireless LAN with 1 Mbps and 2 Mbps communication capabilities. Following the FCC regulations, the DSSS system provides a processing gain of at least 10 dB by chipping the baseband signal at 11 MHz with an 11-chip code. The baseband modulations of Differential Binary Phase Shift Keying (DBPSK) and Differential Quadrature Phase Shift Keying (DQPSK) are used to provide the 1 Mbps and 2 Mbps data rates, respectively [WLAN]. The transmission distance is about 550 meters for 1 Mbps and 388 meters for 2 Mbps (Outdoor)[PR94].

Frequency hopping spread spectrum for the 2.4 GHz ISM band

The IEEE 802.11 frequency-hopping physical layer standard uses 79 non-overlapping 1 MHz channels to transmit a 1 Mbps data signal over the 2.4 GHz ISM band. This band ranges

from 2400 MHZ to 2483.5 MHz, providing 83.5 MHz of bandwidth. A channel hop occurs every 224 μsec. Each IEEE 802.11 network uses a particular hopping pattern that allows up to 26 networks to be allocated [WLAN]. This mode offers lower performance than DSSS and is seldom used in product.

Infrared physical layer

The IEEE 802.11 infrared physical layer operates in the near visible light range of 850 to 950 nanometers. The transmission distance is limited to the range of 10 to 20 meters [WLAN].

2.2.2 IEEE 802.11b

The IEEE 802.11b [WLANB] is a high rate extension of the 802.11 DSSS. It provides 5.5 Mbps and 11 Mbps data rates in addition to the 1 Mbps and 2 Mbps rates defined in IEEE 802.11 DSSS. It uses the Complementary Code Keying (CCK) modulation [WLANB] that provides higher data rate, using the same chip (11MHz) as the basic 802.11 DSSS. The transmission distance is 235 meters for 5.5 Mbps and 166 meters for 11 Mbps (Outdoor)[PR94].

2.2.3 IEEE 802.11a

The physical layer of 802.11a [WLANA] is based on the OFDM,operates at the 5.15–5.25, 5.25–5.35 and 5.725–5.825 GHz unlicensed national information infrastructure (U-NII) bands, and can support data rates as high as 54 Mbps. The OFDM works on the principle of splitting a high stream data rate into a number of lower rate streams, which are transmitted simultaneously by multiple orthogonal subcarriers. Table 2.1 [WLANA] lists the main parameters of the OFDM based IEEE 802.11a standard.

Table 2.1 Main parameters of OFDM

Parameter	Value
Data rate	6,9,12,18,24,36,48,54 Mbps
Modulation	BPSK, QPSK, 16QAM, 64QAM
Coding rate	1/2, 2/3, 3/4
Number of subcarriers	52
Number of pilots	4
OFDM symbol duration	4 μs
Guard interval	800 ns
Subcarrier space	312.5 KHz
Channel spacing	20 MHz
-3 dB Bandwidth	16.56 MHz

2.2.4 IEEE 802.11g

IEEE 802.11g [WLANG] is designed to provide a higher-speed, backward-compatible physical-layer extension to the highly successful IEEE 802.11b standard. It is compatible with the 802.11 MAC, implementing all mandatory portions of the existing IEEE 802.11b standard to ensure compatibility and interoperability, and includes a maximum data rate of at least 20 Mbps when 802.11b nodes are not present to force network to fall back to 802.11b way of functioning.

The IEEE 802.11g standard takes the IEEE 802.11b standard requirements and adds extra rate capabilities to extend the data rate in the 2.4-GHz band up to 54 Mbps. It provides 5.5 Mbps and 11 Mbps using Complementary-Code Keying (CCK). The optional 5.5 Mbps and 11 Mbps Packet Binary Convolutional Coding (PBCC) of IEEE 802.11b has been extended in IEEE 802.11g to 22 and 33 Mbps.

To achieve data rates up to 54 Mbps, the IEEE 802.11g uses OFDM as IEEE 802.11a does. The difference is that IEEE 802.11g uses the same encoding formatting in the 2.4-GHz band to achieve the same rates, mandating the OFDM rates of 6, 12 and 24 Mbps.

An additional optional mode, CCK-OFDM, is included in the IEEE 802.11g standard, which uses the Barker-encoded preamble of 802.11b with an OFDM payload. The CCK-OFDM scheme supports payload rates of 6, 9, 12, 18, 24, 36, 48 and 54 Mbps.

2.3 IEEE 802.11 MAC Sublayer

The MAC sublayer is a set of protocols that is responsible for the allocation of the shared transmission medium. The IEEE 802.11 MAC sublayer uses the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol [WLAN]. It is specified in terms of coordination functions that determine when a station is allowed to transmit MSDUs. There are two functions defined in the IEEE 802.11 wireless LAN: the Distributed Coordination Function (DCF) that is used for asynchronous data transmission and the Point Coordination Function (PCF) that can be implemented at the Access Point to support time bounded traffic.

2.3.1 MAC Frame Format

The MAC frame format for the IEEE 802.11 is shown in figure 2.2.

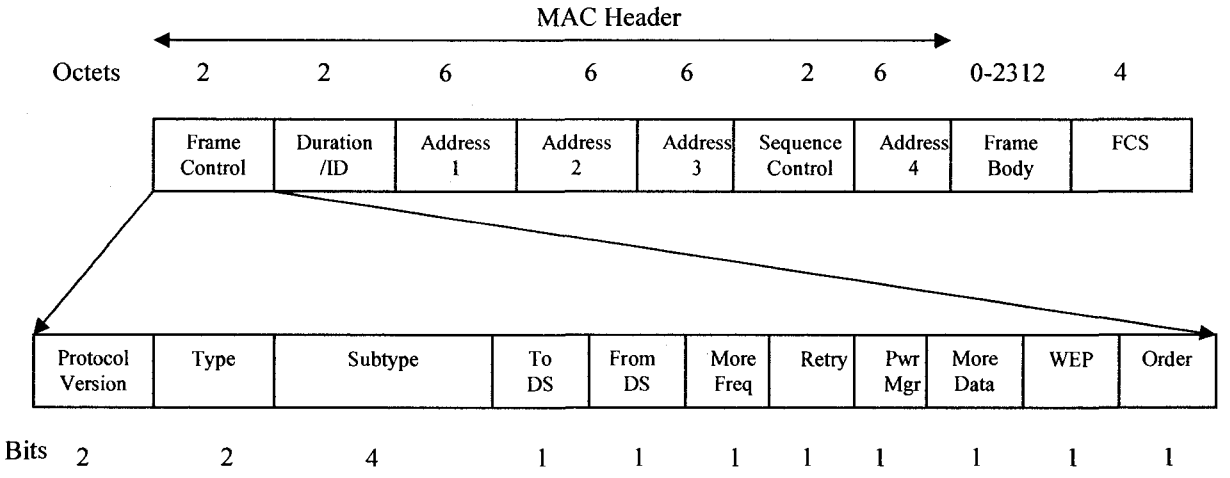


Figure 2.2 MAC frame format

Each frame consists of the following basic components:

- A *MAC header*, which comprises frame control, duration, address, and sequence control information;

- A variable length *frame body*, which contains the information specific to the frame type;
- A *Frame Checks Sequence* (FCS), which contains an IEEE 32-bit Cyclic Redundancy Code (CRC).

2.3.2 Carrier Sense Mechanism

The carrier sense is performed at the physical layer and MAC sublayer. At the air interface, the network card analyzes power strength of the received wireless signal in order to detect if there are other mobile stations that are transmitting. For the MAC sublayer, the standard defines a Network Allocation Vector (NAV) in the header of the data frame, which is used to specify the duration of the MPDU. When one station sends a frame, it uses the NAV to inform other stations in the same BSS how long it will occupy the wireless medium.

2.3.3 Interframe Space

The Interframe Space (IFS) is the time interval between the transmissions of two adjacent frames. The IEEE 802.11 standard specifies three different kinds of IFS intervals as follows in the descending priority order:

- SIFS: Short interframe space = 10μs
- PIFS: PCF interframe space = SIFS + slot time (20μs)
- DIFS: DCF interframe space = PIFS + slot time (20μs)

The usage of the IFS (SIFS, PIFS, DIFS) is determined by the nature of the packet to be transmitted. SIFS is the shortest IFS and is used for those packets that need to have the highest priority to access the wireless medium. Using the SIFS between transmissions within the frame exchange sequence prevents other stations, which are required to wait for a longer IFS (PIFS or DIFS), from attempting to use the medium. SIFS is used for ACK frame, CTS frame, the second or subsequent MPDU of a fragment burst, and by a station answering the polling issued by the access point during PCF.

PIFS is used only by stations operating under PCF, in order to gain accesses to the medium and start CFP period

DIFS is used by stations operating under DCF, in order to transmit data frames (MPDUs) and management frames (MMPDUs).

2.3.4 Distributed Coordinator Function

The IEEE 802.11 standard defines the Distributed Coordinator Function (DCF) as the basic coordination function. It is based on the CSMA/CA [WLAN]. The DCF is able to work in Ad Hoc as well as infrastructure network configurations. The access mechanism of DCF is shown in figure 2.3.

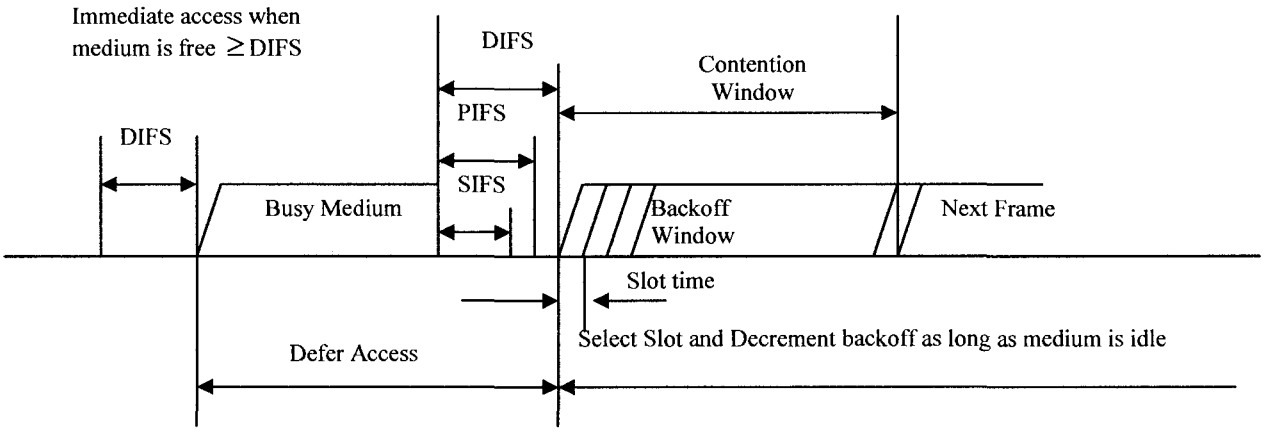


Figure 2.3 DCF basic access method

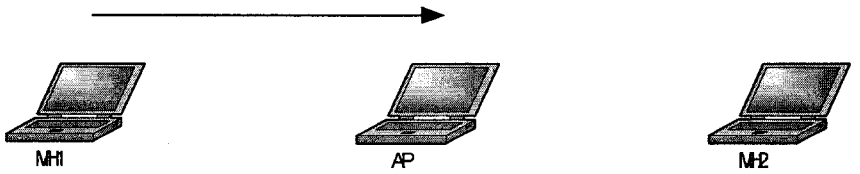
When a station prepares to send a frame, it must keep on sensing the physical medium until the channel becomes idle. Then, it must wait for a DIFS and sense the medium again. If the medium is still idle, the MPDU will be sent out. The target station decides whether the received frame is damaged by calculating the CRC value, and if there are no errors, it sends an ACK frame back to the sender after waiting a SIFS period. When the source station receives this ACK frame, it knows that this transmission was successful and prepares for sending the next packet in queue. If the ACK frame is lost during the transmission, the source station concludes that the previous transmission failed and has to send the frame again.

In order to reduce collisions because two or more stations begin transmitting at the same time, a random backoff procedure is defined in the DCF. The random backoff procedure is based on the binary exponential backoff algorithm. When a station prepares to send out a frame, it waits for a DIFS period, during which the channel is idle. Then, the station calculates a random backoff time and starts the backoff timer. The frame can be sent out only if the medium is still idle after the random backoff period expires. The back off period is called competition period (CP) and is calculated in time slots[1]. However, if the medium becomes busy before the timer expires, the station "freezes" the timer, restarting it only after the medium becomes idle again. If a collision happens during the transmission process, both stations have to retransmit the "packet" and recalculate the backoff time. The size of contention window will double for every failed transmission attempt, until it reaches to maximum value.

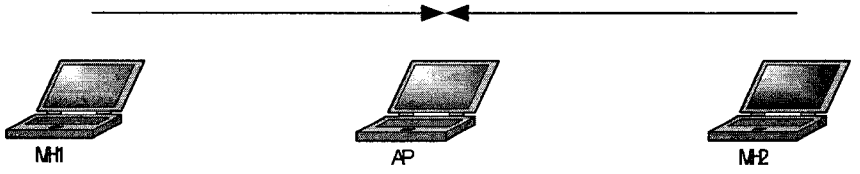
2.3.5 RTS/CTS

In a wireless LAN, the "hidden node" phenomenon happens if one station fails to sense the transmission of an active station. Figure 2.4 provides an example of how the "hidden node" phenomenon occurs. Node A can communicate with node B, and node B can communicate with node C. But node C cannot sense the existence of node A because of the long distance. Thus, collision will happen at node B when node A and node C send packets to B at the same time. To reduce the collisions induced by the "hidden node" phenomenon, the Request-To-Send (RTS) and Clear-To-Send (CTS) control frames are used before the station sends out the MPDU. The procedure is shown in figure 2.5 [WLAN]. Whenever a station wishes to send out a frame, it first sends out short RTS packets containing information regarding the length of the data frame and the expected duration of the transmission. After the destination station receives the RTS, it responds with a CTS packet, which authorizes the transmission, notifies all the stations in the BSS that a transmission will happen and informs other stations to hold off transmission until the current transmission finished.

[1] In 802.11, the time slot value depends on the correspondingly named PHY characteristic.



MH2 can not sense the existance of MH1



MH2 transmits a packet, which collides with the packet sent by MH1 at the AP

Figure 2.4 Hidden node

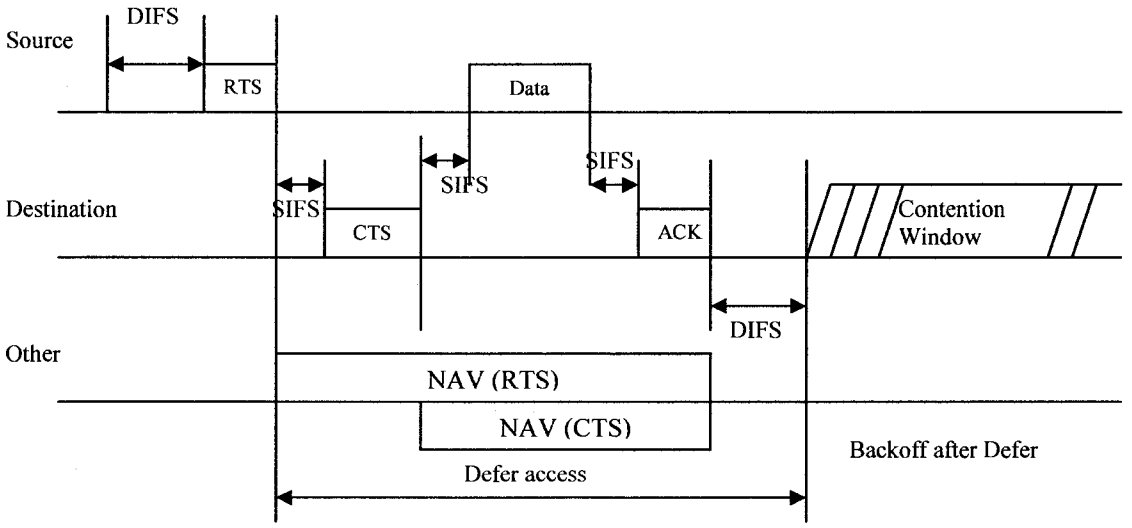


Figure 2.5 RTS/CTS process

2.3.6 Fragmentation

IEEE 802.11 defines a fragmentation and reassembly policy for the purpose of reducing the probability that a MPDU will be damaged in the error prone wireless environment. When a large MSDU is transmitted, the source station compares its frame size to the fragmentation

threshold. If the size of MPDU exceeds the value of fragmentation threshold, it is spited to several smaller fragments with the size of fragmentation threshold. All of these smaller fragments are transmitted one after the other and reassembled at the destination. The functionality of the protocol prevents other stations accessing the wireless medium while there are fragment waiting for transmitting. The wireless medium is released only after the whole MPDU is transmitted successfully. The process is shown in figure 2.6.

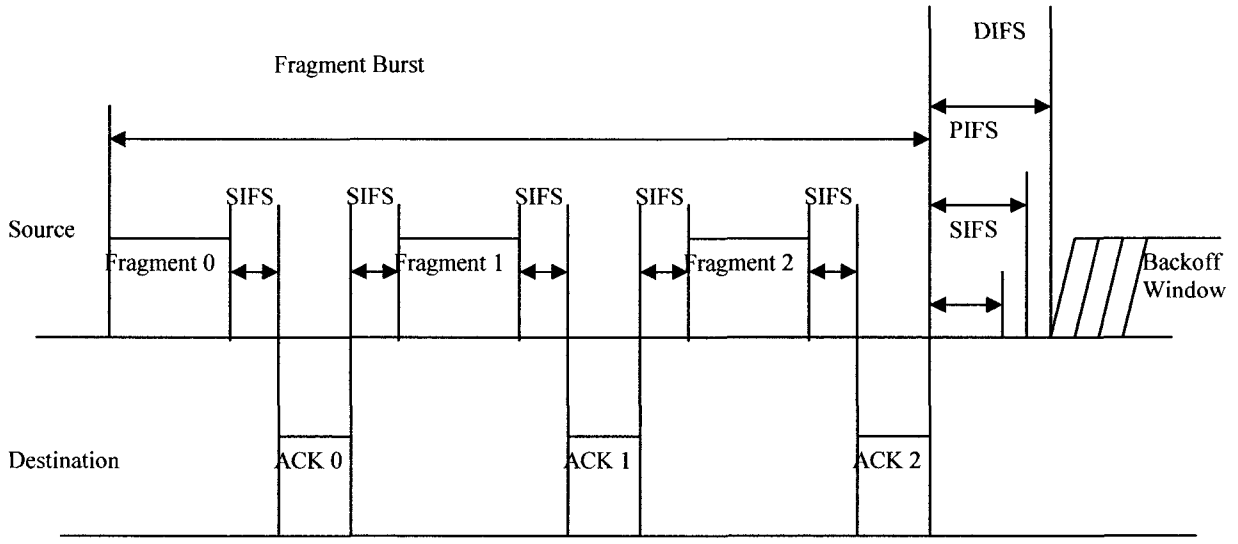


Figure 2.6 Transmission of multi-fragmented MSDU

2.3.7 Point Coordinator Function

The Point Coordinator Function (PCF) provides contention free frame transmission for real-time QoS requiring applications. It is an option supported only in infrastructure networks. Within each BSS, the AP performs the function of point coordinator and uses the station-polling method to avoid contention and collision.

Logically, the PCF works on the top of the DCF. The frequency of occurrence of PCF happening during the DCF procedure is determined by the Contention Free Period Repetition Intervals (CFPrate), which is initialized by a beacon frame sent out from the AP. The AP also determines the duration of Contention Free Period (CFP).

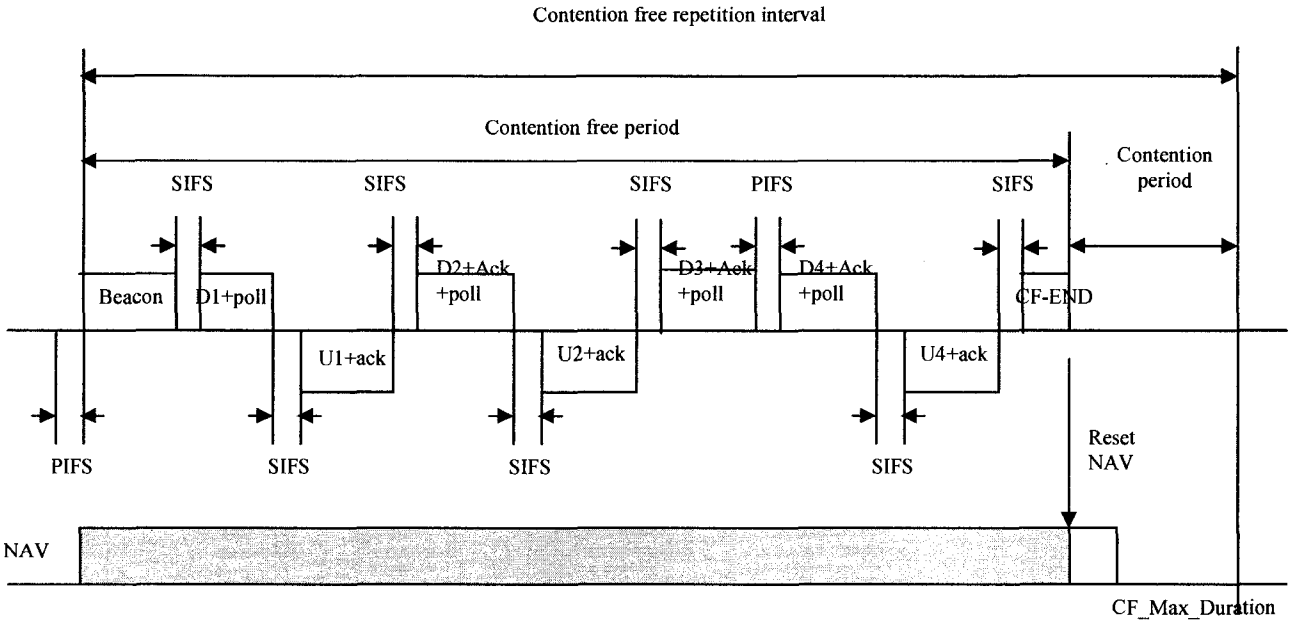


Figure 2.7 PCF frame transfer procedure

The PCF procedure is shown in figure 2.7. At the starting point of the CFP, the AP, where the point coordinator is running, first senses the wireless medium. If the channel remains idle for a PIFS time period, a PCF initial beacon frame is sent out by the coordinator function to start the CFP. After a SIFS interval, the AP sends out a poll packet to the first station in the polling list. After the destination station receives the polling packet, it returns a CF-ACK or DATA+CF-ACK, after waiting for a SIFS interval. The AP then sends a polling frame to the next station following the polling sequence list. During the CFP, a station transmits frames only after the polling frame with its address is received. The coordinator function can terminate the CFP by sending out a CFP end (CFend) frame to inform all the stations that the CFP is finished and the stations can begin competing for the wireless transmission medium again. In order to improve the efficiency, the polling frame and the ACK frame can be inserted into data frames to reduce the overhead. If the point coordinator does not receive the ACK from the polled station, the next station in the polling list will be polled after a PIFS interval.

Through the proper implementation of the polling sequence, the PCF can provide QoS support to throughput and delay sensible applications.

2.4 Summary

In this chapter, we briefly introduced the basic components of the IEEE 802.11 architecture. Several physical layers were presented and the MAC procedures, which support the asynchronous MSDU delivery services, were described.

Chapter 3

Quality of Service Architecture

This chapter presents a brief overview of the integrated services and differentiated services. They are described in Section 3.1 and Section 3.2, respectively. In Section 3.3, several schemes and scheduling algorithms for the QoS enhancement of the IEEE802.11 wireless LAN are introduced.

3.1 Integrated Services

In 1994 the Integrated Services Working Group of the Internet Engineering Task Force (IETF) proposed the Integrated Services (InterServ) architecture [BRA94] as an extension to the existing Internet protocols, in order to support real-time services. It is based on the per-flow reservation of resources in all the routers along the path from the traffic source to the destination. A reservation protocol, the Resource Reservation Protocol (RSVP) [BRA97], was defined to provide integrated services through the Internet. The RSVP is a signaling protocol and is used by a host to request a specific QoS from the network. Routers also use RSVP to deliver QoS requests to all nodes along the path(s) of the flows and to establish and maintain the state table of the traffic flows. Using this method, resources are reserved in each node along the data path.

RSVP operates on top of IPv4 or IPv6. RSVP is designed as an Internet control protocol to operate with unicast and multicast routing protocols. The RSVP reference model is shown in figure 3.1.

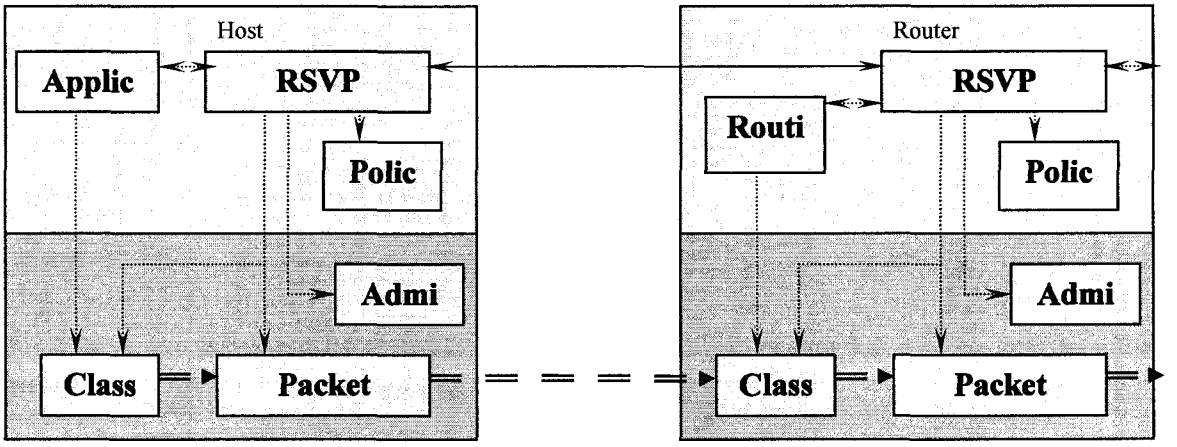


Figure 3.1 RSVP in Hosts and Routers

QoS is implemented for a particular data flow using traffic control mechanisms, which include the packet classifier, the admission control, policy control and the packet scheduler. The functions for each mechanism are as following:

- Packet classifier: Determines the QoS class for each packet
- Admission control: Determines whether the node has sufficient available resources to supply the requested QoS.
- Policy control: Determines whether the user has administrative permission to make the reservation.
- Packet scheduler: Achieves the promised QoS by scheduling the packets according to the QoS class of the packet.

The connection is established through the following steps:

1) An application, running on the receiver host, passes a QoS request to the local RSVP process.
2) The RSVP process sends the reservation request message to the sender along the reverse data path(s) from the receiver to the sender host.
3) At each intermediate node, a reservation request triggers two general actions:

Make a reservation on a link: The RSVP process passes the request to local admission control and policy control. The admission controller module determines

whether the node has sufficient resources available for the requested QoS and the policy control module determines whether the user has administrative permission to make the reservation. If either test fails, the reservation is rejected and the RSVP process returns an error message to the appropriate receiver(s). If both succeed, the node sets the packet classifier to select the data packets defined by the filter spec, and it interacts with the packet scheduler to obtain the desired QoS

Forward the request upstream: A reservation request is propagated upstream towards the sender host.

There are two fundamental RSVP message types: Resv and Path. Each receiver sends RSVP reservation request (Resv) messages upstream towards the senders. These messages follow exactly the reverse path the data packets will use. Going upstream they create and maintain "reservation state" in each node along the path(s). When Resv messages finally reach the senders, each sender host transmits RSVP Path messages downstream along the unicast/multicast routes provided by the routing protocol(s), following the paths of the data. These Path messages store "path state" in each node along the way.

The RSVP protocol mechanism provides a general facility for creating and maintaining the state table of the distributed reservation across a mesh of multicast or unicast delivery paths. *RSVP* takes a "*soft state*" approach where soft state is created in routers and hosts which are periodically refreshed by *Path* and *Resv* messages. The state table is deleted if no matching refresh messages are received before the expiration of a "*cleanup timeout*" interval or by a "*teardown*" message after the applciation terminates.

The main drawbacks of the InterServ architecture are the large overhead for establishing a connection and scalability problems, as each interior node needs to implement per-flow classification and packet scheduling for a very large number of flows.

3.2 Differentiated Service

With the explosive development of the Internet, a scalable QoS architecture is needed. IETF proposed the differentiated service (DiffServ) [BLA98] architecture, which achieves

scalability by implementing the complex classification and conditioning functions only at network boundary nodes, and by applying per-hop behaviors to aggregate traffic flows.

The architecture of differentiated services is based on a simple model. When the traffic enters a network, they are classified at the boundaries of the network and assigned to different behavior aggregates (class). Each behavior aggregate is identified by a single Differentiated Service Code Point (DSCP). Within the core of the network, packets are forwarded according to the per-hop behavior, associated with the DS codepoint.

3.2.1 Differentiated Services Domain

A DS domain is a contiguous set of DS nodes that operate within a well-defined boundary. A common service provision policy and a set of Per-Hop Behavior (PHB) groups are implemented at each node in the DS domain. The boundary nodes of the DS domain classify ingress traffic streams to ensure those packets are appropriately marked. The boundary nodes within the DS domain apply to a packet the forwarding policy, which corresponds to the PHB that is indicated by the DSCP value of the packet. Traffic enters a DS domain at a DS ingress node and leaves a DS domain at a DS egress node.

3.2.2 Traffic Classification and Conditioning

The packet classification policy identifies the subset of traffic streams that may receive a differentiated service. The traffic streams are conditioned and mapped to one or more behavior aggregates and are marked within the corresponding DS codepoint.

Traffic conditioning performs metering, shaping, policing and/or re-marking to ensure that the traffic streams entering the DS domain conform to the rules specified in the Traffic Categories Aggregate (TCA) and the domain's service provisioning policy. The extent of traffic conditioning is dependent on the specifics of the service offering, and may range from simple codepoint re-marking to complex policing and shaping operations. The structure of packet dispatcher and classifier is shown in figure 3.2, in block diagram form.

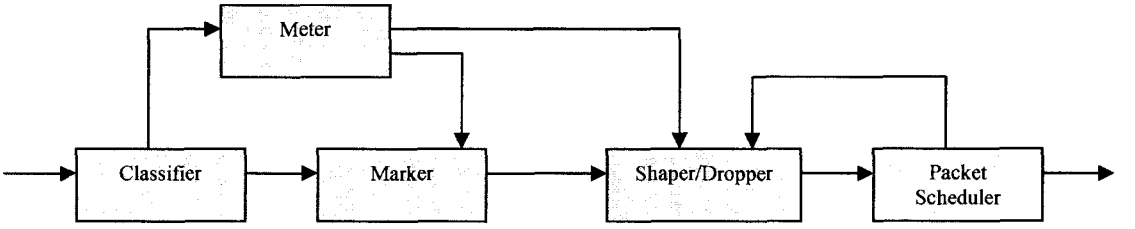


Figure 3.2 Packet classifier and traffic conditioner

3.2.3 Per-Hop Behaviors

A certain per-hop behavior corresponds to a specific externally observable packet forwarding behavior of a DS node that is applied to a particular DS behavior aggregate. IETF has defined two PHBs: Assured Forwarding PHB (AF) [HEI99] and Expedited Forwarding PHB (EF) [JAC99].

3.2.3.1 Assured Forwarding PHB (AF)

The AF PHB [HEI99] is specified to offer different levels of forwarding assurance to IP packets. AF PHB defines four AF classes. Each AF class in a DS node is allocated certain amount of resources (such as buffer space and bandwidth) by the ingress node of the provider's DS domain. Each class offers three possible drop precedence values; thus each class has three sub-classes. In case of congestion, packets with higher drop precedence value will be discarded first to protect packets with lower drop precedence value.

Therefore, in a DS node, the level of forwarding assurance offered to an IP packet depends on:

- The amount of forwarding resources which have been allocated to the AF class to which the packet belongs.
- The current load of the AF class.
- The drop precedence value applied to the packet.

3.2.3.2 Expedited Forwarding PHB (EF)

The EF PHB [JAC99] is used to provide a low loss, low latency, low jitter and end-to-end service through a DS domain, which appears to the endpoints like a point-to-point connection or a "virtual leased line".

The EF PHB is defined as a forwarding treatment for a particular Diffserv aggregate where the departure rate of the aggregate's packets from any Diffserv node must be equal to or exceed a configurable rate. The rate provided to the EF traffic should not be affected by the intensity of any other traffic passing through the same node.

3.3 QoS Enhancement on the IEEE 802.11 Wireless LAN

The increasing popularity of the IEEE 802.11 standard based wireless LAN and the need to support multimedia applications generate the necessity to incorporate Quality of Service (QoS) into this technology. In recent years, several QoS enhancement mechanisms have been proposed for the IEEE 802.11 wireless LAN. In Section 3.3.1, a draft, proposing a QoS capable MAC protocol known as IEEE 802.11e, is presented. In Sections 3.3.2, and 3.3.3, we introduce two distributed scheduling algorithms that provide fair scheduling among traffic flows by modifying the existing IEEE 802.11 MAC protocol. In Sections 3.3.4 and 3.3.5, the long-term channel state based hierarchical wireless link-sharing model and the stateless wireless Ad Hoc network architecture are briefly described.

3.3.1 IEEE 802.11e

The purpose of IEEE 802.11 Task Group E is to enhance the current 802.11 MAC protocol, by expanding support for applications with Quality of Service requirements. Since the task group was initiated in 1999, two new models have been proposed: an enhanced version of DCF (EDCF) and a Hybrid Coordination Function (HCF).

3.3.1.1 Enhanced DCF

EDCF [MAN02] provides differentiated services on the MAC sublayer by introducing different Traffic Categories (TCs). A station can have as much as eight TCs to map the differentiated QoS parameters. In the Contention Period (CP), each TC contends for the

transmission opportunity (TXOP) independently and begins the backoff process after the channel has been idle for an Arbitration Interframe Space (AIFS). The values of the AIFSs are different for the eight TCs in order to reflect the different QoS requirements. The minimum value of the AIFS equals to the DIFS, which belongs to the TC with the highest priority. Another parameter, which is dependent on the TC priority, is minimum the contention window (CW_{min}). The TC of a higher priority has a smaller CW_{min} value, which means that packets corresponding to this TC wait for a shorter period than packets of TCs of lower priority during the backoff process. Thus, a packet that belongs to TC of higher priority has better opportunity to get the TXOP. The EDCF mechanism is shown in figure 3.3.

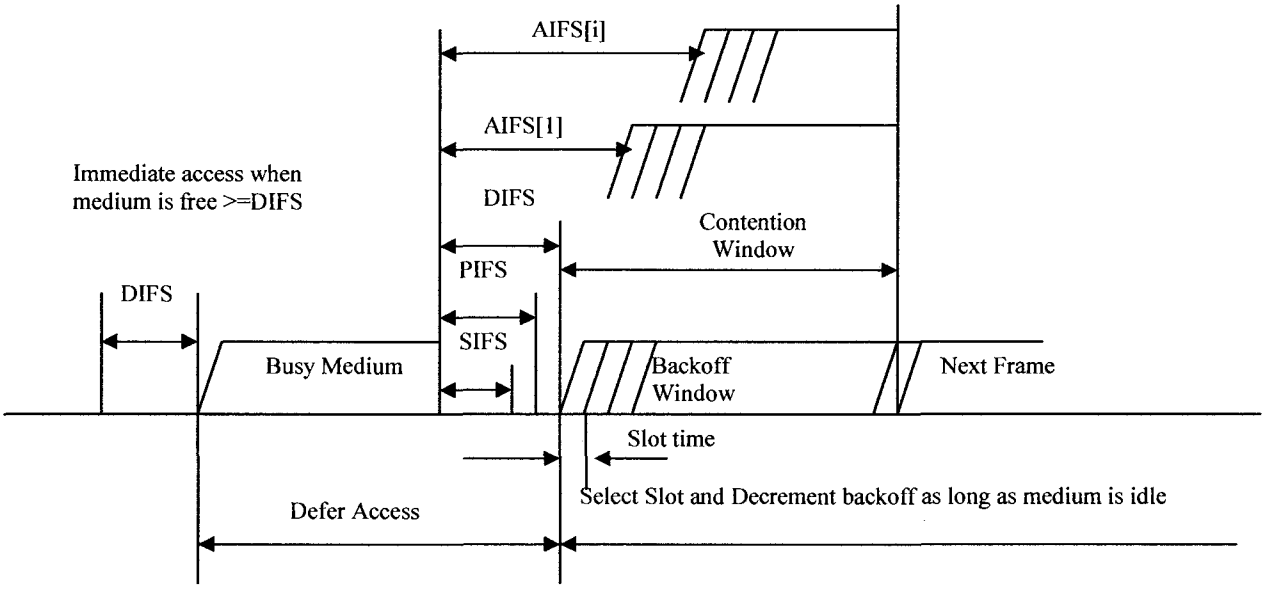


Figure 3.3 EDCF mechanism

3.3.1.2 Hybrid Coordinator Function

Based on the EDCF and the PCF, the HCF can provide QoS support to time-bounded traffic streams by using the contention free and controlled contention transmissions during the CFP and CP. HCF implements a polling scheme which is controlled by the Hybrid Coordinator (HC). The typical IEEE 802.11e HCF superframe is shown in figure 3.4 [MAN02].

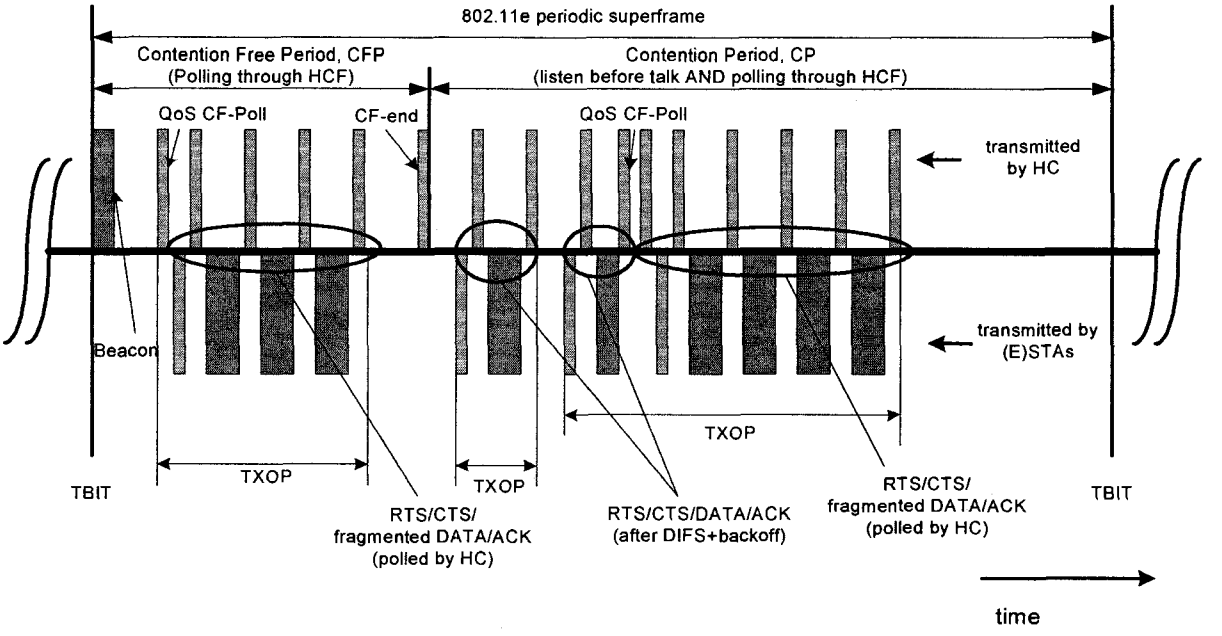


Figure 3.4 HCF superframe

The HC has a higher priority to access the transmission medium than the EDCF and DCF because HC uses the PIFS, which is shorter than the DIFS and AIFS that are used by the DCF and EDCF. The HC starts a CFP by sending out a beacon, in which starting time of the CFP and the maximum duration of the TXOP are encapsulated. During the CFP, the HC sends QoS CF-Poll frames to stations according to a sequence that reflects the QoS priority of those stations. Each station tries to access the transmission medium only after it receives the QoS CF-Poll frame from the HC and uses the SIFS as the value of the inter frame space. The HC can terminate the CFP at any time, by sending out a CF-End beacon frame. A new feature of the HCF is that the HC can send the CF-Poll frame to the stations during the CP. During the CP, each station gets the TXOP after it waits for AIFS plus the backoff time or after it receives the QoS CF-Poll frames. In order to make the HC get the updated QoS requirements of the stations, a controlled contention mechanism is used to allow the stations to request polling frames by sending out resource request frames.

3.3.2 Distributed Weighted Fair Queuing

The Distributed Weighted Fair Queuing (DWFQ) [BAN02] is a fully distributed QoS scheme, which modifies the DCF mode of the IEEE 802.11 MAC protocol, extending it to

distribute the bandwidth of the wireless LAN among different traffic flows proportionally to their weights.

With the weighted fair queuing [ABH93], every flow in the flow set is assigned a weight. Here we consider that flow *i* belongs the flow set A. The flow set A has N flows and the weight of flow i equals to w_i. The weight w_i represents the share that flow *i* can obtain from the total bandwidth of the flow set A. The bandwidth r_i, received by the individual traffic flow i, is proportional to the weight w_i to which the traffic flow has been assigned:

$$\frac{r_j}{w_j} = \frac{r_i}{w_i} \qquad \forall i, \forall j \quad (1 \le i, j \le N) \tag{3.1}$$

where the default weight value is set to 1 and corresponds to the basic service. In this paper, the default *weight* equals to 1. This value corresponds to the basic service; *weights* smaller than 1 are not allowed, and any larger *weight* means "better than average" kind of treatment.

The DWFQ implements the WFQ by assigning different Contention Window (CW) sizes to the mobile stations with different weights based on the following observation: in the DCF, the bandwidth received by a mobile station is closely related to the size of the CW. The smaller the size of the CW, the higher the probability of the station to access the medium becomes, the larger the bandwidth the station receives becomes as well.

In order to properly adjust the CW size, the DWFQ defines a variable for flow *i*, which is named label L_i. The relation between L_i, r_i, w_i is the following:

$$L_i = \frac{r_i}{w_i} \tag{3.2}$$

The bandwidth r_i is updated every time a new packet of flow i is transmitted:

$$r_i^{new} = (1 - e^{-t_i/k}) \frac{l_i}{t_i} + e^{-t_i/k} r_i^{old} \tag{3.3}$$

where l_i and t_i are the length and inter-arrived time of the transmitted packets. k is a constant and is set to 100ms [STO98].

The bandwidth allocation expressed in (3.1) is achieved when the label values of all the flows become equal:

$$L_i = L \qquad \forall i \tag{3.4}$$

The actual value of L is dependent on the different implementation scenarios (for example, it is dependent on the number of flows).

The DWFQ algorithm is implemented based on the DCF of the IEEE802.11 MAC protocol. Each station calculates the label value independently and encapsulates the value in the header of the packets it sends. At the same time, it keeps on checking the label values of other stations. When one station finds out that its L_i value is smaller than the L_i values of other stations, it increases its CW by a small amount, while in the opposite case it decreases its CW by a small amount. This way, the L_i values of all flows tend towards a common value L and the bandwidth resource can be allocated according to the weights of the flows.

The following issues are considered in the adjustment of CW size:

- The CW should not be increased above the values defined by the 802.11 standard; for the reason of maintaining backwards compatibility, the basic service (with a *weight* equal to 1) uses the CWs defined in the 802.11 standard, and any higher *weight* should receive a "better than average" kind of treatment and gets a smaller CW value.

- If transmitting below the desired rate happened because of the low sending rate (for example, the application only require a low data rate), the CW should not be decreased. This can be detected by the fact that in this situation the transmission queue is empty

- CWs should not be allowed to decrease if they negatively influence the overall performance of the network. If the channel is detected to be below its optimum limit

of throughput due to too small values for the CWs (i.e. overload), the CW should be increased.

3.3.3 Distributed Fair Scheduling

The distributed fair scheduling (DFS) [VAI00] is designed to emulate the Self-Clocked Fair Queuing (SCFQ) [GOL94] in a distributed manner by modifying the DCF of the IEEE802.11 wireless LAN MAC protocol.

Similarly to the SCFQ, the DFS transmits the packet with the smallest finish tag and the virtual time is updated to be equal to the finish tag of the most recently transmitted packet. A distributed approach for determining the smallest finish tag is employed by using the modified *backoff interval* mechanism of the DCF from the IEEE 802.11 MAC. The DFS implements an algorithm to calculate the backoff interval that is proportional to the finish tag of the packet to be sent.

The backoff interval B_i of the flow i for the k_{th} packet arriving at the head of the out-queue is calculated as:

$$B_i = \left[Scaling \quad \_ \; Factor \quad * \frac{L_i^k}{w_i} \right], \tag{3.5}$$

where the *Scaling_Factor* is a suitable scale for the virtual time, the L_i^k represents the length of the k_{th} packet and ϕ_i is the weight of flow i.

There is a trade-off between throughput and fairness achieved by the choice of *Scaling_Factor*. In [VAI00], the authors investigated the influence of the Scaling_Factor. The results show that throughput initially increases when the *Scaling_Factor* is increased, but then degrades after the *Scaling_Factor* is increased further. A small *Scaling_Factor* results in too many collisions, which reduces the throughput. However, the large backoff interval, which results from the larger *Scaling_Factor*, leads to a bigger overhead if the *Scaling_Factor* is increased to much.

In order to reduce the collision probability when more than one stations count down to 0 simultaneously, a random number p (uniformly distributed in [0.9, 1.1] as the author used in the simulation of this paper) is used to calculate the actual backoff interval B_a:

$$B_a = p * B_i \tag{3.6}$$

With the combined use of the packet length and the flow weight in the calculation of the size of backoff intervals, the traffic flows with different weights are assigned different priorities to access the transmission medium in DCF.

3.3.4 Long-term Channel State based Hierarchical Wireless Link-sharing Model

The distributed QoS mechanisms described in the previous sections implement fair queuing scheduling by modifying the DCF of the IEEE 802.11 MAC protocol. However, this requires the development of new Network Interface Cards (NICs). The long-term channel state based hierarchical wireless link-sharing model [WIS01] proposed an alternative method, able to provide QoS support on the legacy IEEE 802.11 MAC protocol. This mechanism uses the long-term quality ("the average quality") of the wireless link towards a specific mobile host. Although this information is less time-critical and updated in relatively large intervals, this wireless scheduling can be implemented above the link layer and is able to improve the controlled sharing of the wireless link.

The long-term wireless channel quality is measured using the channel state monitor, which is developed by the author based on Linux. Because there is no support for monitoring wireless channel state linked to a specific mobile host existed in current wireless LAN hardware, the wireless tool extension [TOU] is used to get the signal level/quality for wireless LAN card. A calibration process is also needed in order to get the long-term wireless channel quality/date rate mapping for one specific station. The process includes the following steps:

- Deactivate all mobile stations except one.
- Start calibration tool on mobile station and packet generation utility on access point.
- Record the data rate and channel signal quality data in a large variety of positions.
- Make the long-term channel quality/date rate mapping table

This scheduling algorithm defines the term *goodput* as the amount of data seen above the link layer, whereas the term *raw throughput* is defined as the capacity of the wireless link. Usually we calculate the throughput by counting the total number of bits received in the unit time and use this value in the scheduling algorithm. The is fair for the wired line LAN since the error probability is low and all stations experience the same link quality, in which the goodput is equal to throughput. In the case of the wireless LAN, where a large number of retransmissions happen because of the higher error probability of the wireless channel, the goodput and throughput are not equal anymore. If the scheduling algorithm still uses the throughput, it will lead to wrong results.

For example, when a hierarchical link-sharing scheduler, such as Class Based Queuing (CBQ) [FLO95] or the Hierarchical Fair Service Curve Algorithm (H-FSC) [STO97] is deployed above the link layer, the scheduler will not see any link layer retransmissions and assumes that the goodput is equal to the raw throughput. Therefore, in the case of bad wireless channel conditions, the equal goodput of two mobile stations can lead to completely different shares of the consumed raw throughput.

In order to resolve this unfairness, the goodput to throughput ratio (GTR) is used in this mechanism, which is defined as:

$$g_i = \frac{b_{good,i}}{b_{raw,i}} \tag{3.7}$$

g_i is the GTR, $b_{good,i}$ is goodput and $b_{raw,i}$ is throughput. The packet scheduler above the link layer uses this GTR value to estimate the used raw bandwidth $b_{raw,i}$ when it sends data to a specific destination. Here assumes that scheduling is done for N mobile stations. $b_{good,i}$ is the rate of goodput scheduled for a mobile station i and B_{raw} corresponds to the total available resources of the wireless channel. The amount of available total goodput $B_{good}(t)$ at time t is distributed according to the weights w_i of the mobile stations. Thus, if a class i is in the set of active classes

$$\frac{b_{good,i}(t)}{B_{good}(t)} = \frac{w_i}{\sum_{j \in A(t)} W_j} \quad (i, j \in A) \tag{3.8}$$

Therefore the total amount of available goodput $B_{good}(t)$ is

$$B_{good}(t) = \frac{B_{raw}}{\sum_i \frac{W_i}{g_i(t)_i}} \quad (i \in A) \tag{3.9}$$

and the goodput scheduled for a single station i is:

$$B_{good,i}(t) = w_i \frac{B_{raw}}{\sum_i \frac{W_i}{g_i(t)_i}} \quad (i \in A) \tag{3.10}$$

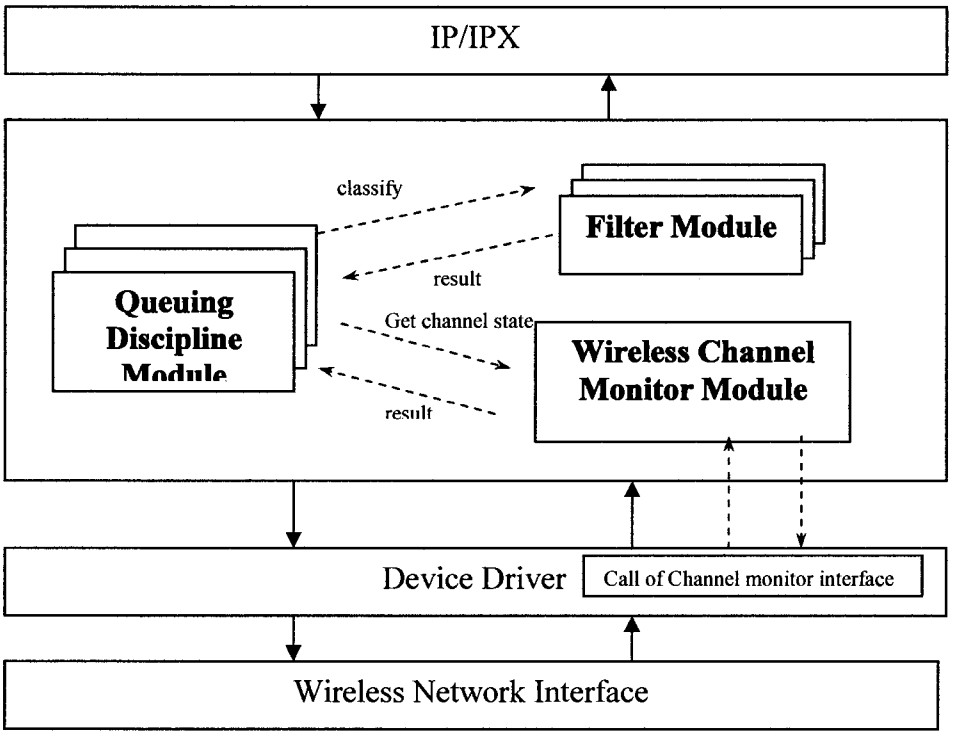


Figure 3.5 The wireless scheduler extension on Linux traffic control

Figure 3.5 [WIS01] shows the implementation of this long-term channel state based on the hierarchical wireless link-sharing model on the Linux traffic control. The *wireless channel monitor* is designed as a loadable kernel module similar to the qdisc and filter/policer

modules. It provides the wireless channel quality information (measured by iwconfig/iwspy tools) via a new interface towards the wireless qdisc.

This technique has the following features:

- Support for the implementation of a wide range of wireless scheduling algorithms.
- Configuration of wireless scheduling components with the same mechanisms as those used in wire-line setups (such as tc, netlink socket).
- Separate channel monitoring and scheduling functionality.
- Minimization of the necessary modifications of hardware.

The disadvantage of this scheme is the requirement of a calibration process. The process is needed to generate a wireless channel quality/goodput mapping for a specific wireless card in a specific environment, which makes it is very difficult to use this scheme in the dynamically changing wireless networks.

3.3.5 Stateless Wireless Ad Hoc Networks (SWAN) Architecture

SWAN [AHN02A], [AHN02B] operates on a best effort MAC and uses feedback-based mechanisms to support soft real-time services and service differentiation in mobile ad hoc networks. The model of SWAN is shown in figure 3.6 [AHN02A].

SWAN applies a distributed approach to prevent congestion. The rate control of best effort TCP and higher priority UDP traffic is performed at every mobile node in a fully distributed and decentralized manner. A rate control is applied to restrict best effort traffic, yielding the necessary bandwidth that is required by real time traffic streams. Adaptive rate control also allows the best effort traffic to efficiently utilize the bandwidth that is not used by the real time traffic at a certain time. The total rate of all best effort traffic and real time traffic transported over each local shared media channel should be maintained below a certain "threshold rate", which prevents delays from becoming excessive.

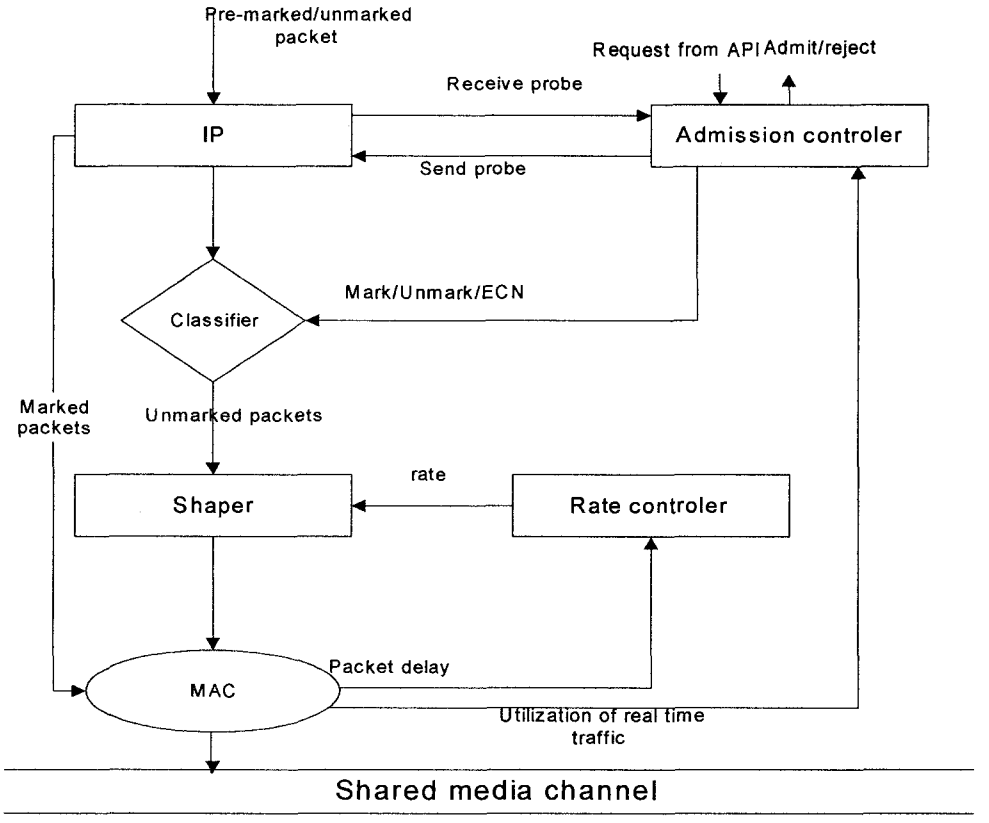


Figure 3.6 SWAN model

```
Procedure update_shaping_rate()
/* called every T second period */
Begin
If(n>0)                    /* one or more packets have delays
                              bigger than the threshold d seonds */

s ← s*(1 − r/100)          /* multiplicative decrease by r% */

else

s ← s + c                  /* addictive increase by c Kbps */

if( (s − a) > a*g/100 )    /*difference between actual rate and
                             shaping rate is bigger than g% of actual rate */

s ← a*(1 + g/100)          /* adjust shaping rate to match actual rate */
```

Figure 3.7 SWAN AIMD rate control algoritm

SWAN adopts the well-known Additive Increase Multiplicative Decrease (AIMD) rate control mechanism [CHIU89], which is shown in figure 3.7. Each mobile host increases its transmission rate gradually (additive increase with increment rate of c Kbps every T seconds), until the packet delays become excessive. When one or more packets have greater delays than the threshold delay d sec, which is based on the real-time delay requirements of applications in wireless network, the rate controller backs off the rate (multiplicative decrease by r percent).

A classifier and a shaper operate on the interface between the IP and MAC layers. The function of classifier is to separate real time and best effort packets, passing the second to the shaper. The shaper operates on the principle of the leaky bucket algorithm. The goal of the shaper is to process best effort packets according to the rate calculated by the AIMD rate controller.

The SWAN architecture does not maintain flow or session state information at intermediate nodes. Furthermore, there is no admission control decision taken at intermediate nodes. Only at the source node the admission controller will decide whether to admit a new session or not. The admission controller, which is implemented at every mobile device, estimates the available local bandwidth by probing the network between the source and destination by using a probing request packet, which is a UDP control packet containing a "bottleneck bandwidth" field. The process for admitting a new real-time session is as follows:

1. The admission controller located at the source node sends a probing request packet towards the destination node in order to estimate the end-to-end bandwidth availability.
2. Each intermediate node between the source-destination pair intercepts the probing request packet and updates the bottleneck bandwidth field in the packet if the bandwidth availability at the node is less than the current value of the field.
3. The destination node sends a probing response packet back to the source node with the bottleneck field copied from the probing request message received by the destination node.

Based on the results of a request/response probe, the session admission controller makes a decision to admit a new session as a real time flow or not. SWAN uses the DS (DiffServ) field to identify the class to which the packet belongs.

Chapter 4

Distributed Network Congestion Control QoS Scheme

In order to provide QoS support using the currently commercially available IEEE 802.11b wireless LAN products, we propose a distributed network congestion control scheme, which is implemented on top of the "best effort" DCF function. In this chapter, the system architecture and functional description of the QoS scheme are presented. The details of its implementation on Linux Red Hat 7.3 are described in the next chapter.

4.1 Introduction of the Network Architecture

A wireless network architecture, capable of covering wide areas, is displayed in figure 4.1. In this architecture, the access points are connected through a backbone network. This could be based on wired or wireless networking technologies. Multiple wireless LAN subnets are connected through this network. Our objective is to establish QoS support to the local footprints by using readily available WLAN hardware.

The algorithms required for the provision of the QoS support are located at the Access Point and the user devices. We propose a QoS scheme capable of providing different classes of service (EF and Best Effort). This QoS scheme is developed on top of the IEEE 802.11 DCF function. This allows its easy implementation using off-the-shelf wireless LAN products. As the reader will find out, it only requires that a new patch is added to the WLAN card driver software. The QoS scheme is able to provide service differentiation among the traffic streams at both directions (downlink and uplink traffic).

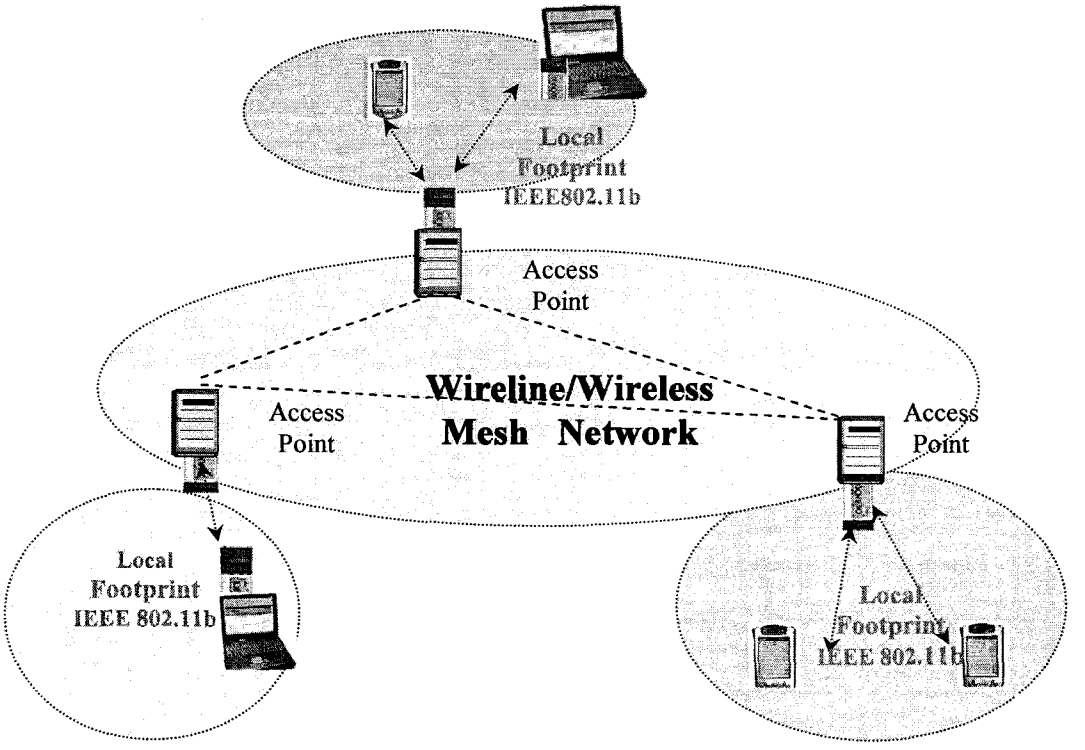


Figure 4.1 The QoS capable wireless LAN network architecture

4.2 Distributed Network Congestion Control Scheme

The objective of this QoS scheme is to provide EF/AF/BE service differentiation capabilities to the IEEE 802.11 wireless LAN. Unlike the wired data link technologies that provide a fixed service capacity, a wireless IEEE 802.11 data link adapts its core throughput capacity according to the external conditions, such as transmission path, signal level, radio noise etc. For example, when a mobile user moves away from an Access Point and the signal strength fades, an IEEE 802.11b wireless card can reduce its data link rate from 11 Mbps down to 5.5 Mbps, 2 Mbps or even 1 Mbps, in order to accommodate the situation. This change of channel capacity is another serious challenge when provisioning and managing QoS in wireless networks.

In order to resolve this problem while making use of the "Best Effort" MAC layer (using DCF), we propose a distributed network congestion control scheme, which implements a local peer-to-peer network, delay feedback based, rate control mechanism.

4.2.1 Throughput and Link Layer Delay Behavior

Figure 4.2 [CHIU89] describes the general behavior of a network congestion control mechanism. The goal of a congestion control scheme is to maintain the traffic load under the "delay knee" of the link's throughput curve (figure 4.2), thus achieving close to maximum data rate while maintaining short delay.

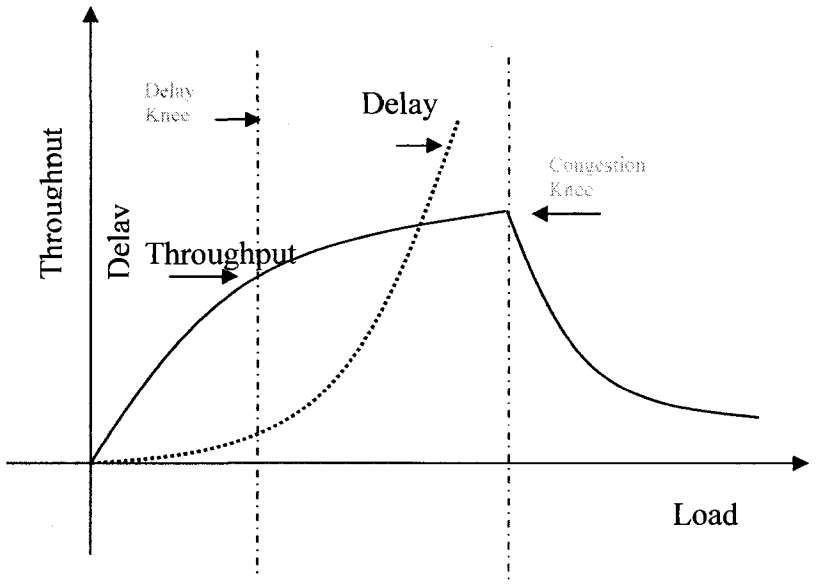


Figure 4.2 The general behavior of throughput and delay versus traffic load for a random multiple access network

The system structure of the proposed distributed network congestion control scheme is shown in figure 4.3. This scheme uses stateless admission control and distributed resource reservation mechanisms to constrain the traffic flows under a threshold of a given link's maximum capacity. At the same time, the rate control mechanism applied on AF/BE traffic is distributed across the networks, allowing each node to adapt on the per-link delay. Following this approach, AF/BE traffic streams can be throttled when bad wireless conditions reduce a link's bit rate or expanded to utilize uncommitted capacity as it occurs.

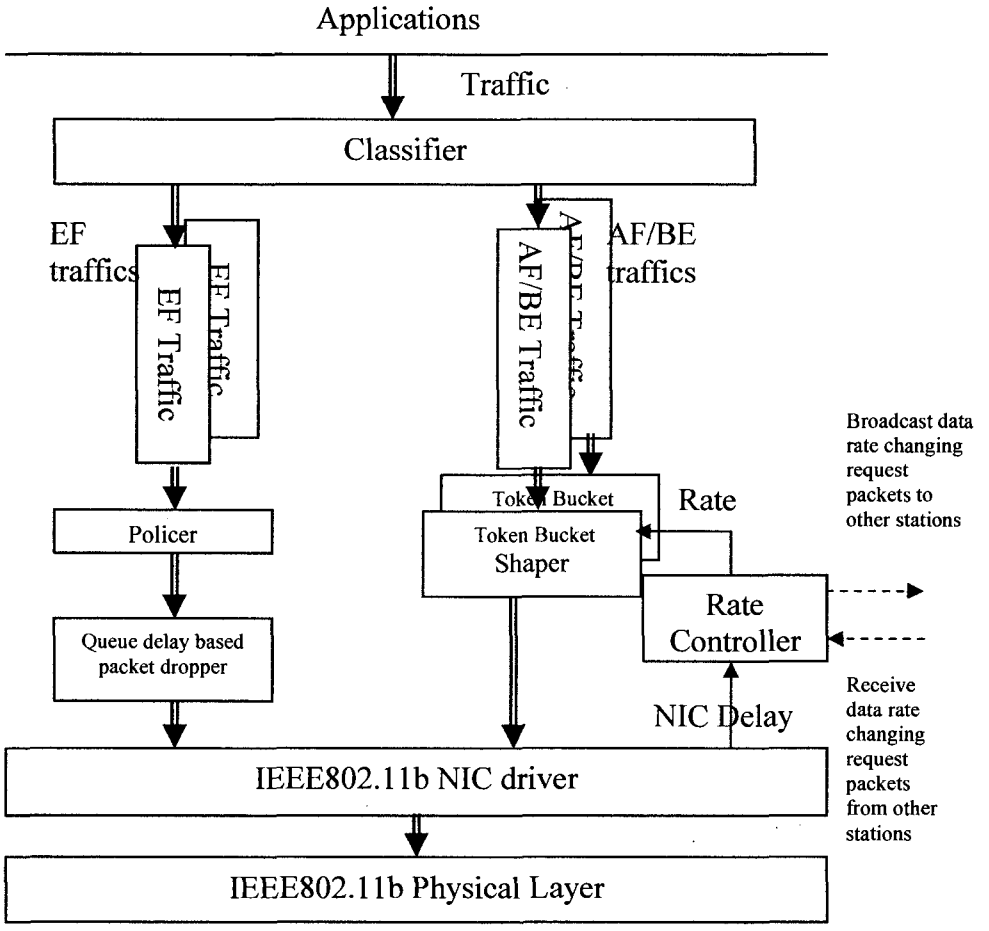


Figure 4.3 The system structure of the distributed network congestion control

In section 3.3.6, SWAN [AHN02A], [AHN02B] was introduced. SWAN operates on a best effort MAC and uses a packet delay feedback-based mechanism to support soft real-time services and service differentiation in mobile ad hoc networks. The rate control of TCP and UDP best effort traffic is performed at every mobile node in a fully distributed and decentralized manner. In order to achieve the bandwidth and delay requirements for the real time UDP traffic streams, the rate control of SWAN needs to know the MAC layer transmission delay, which is not provided by current wireless network cards. In the implementation of the SWAN using commercial available wireless card, the designer used the packet transmission delay, which was measured at the network layer, in order to estimate the MAC transmission delay. However, since the packet delay cannot indicate the real congestion status at the MAC layer, SWAN needs a longer time to adjust the data rate and

resolve the traffic congestion. So the dependency on the MAC delay makes it difficult to use SWAN in the commercial available wireless products.

On the contrary to SWAN, we proposed the distributed network congestion control QoS scheme based on the NIC delay, which can be measured in the Linux driver of IEEE 802.11 wireless card without modifying the current hardware.

4.2.2 NIC delay

Our method follows a different approach. Instead of observing the MAC delay, we measure what we call "NIC delay". The NIC delay is the time it elapses, from the moment a packet is passed from the network layer to the driver of the wireless card, until the moment the driver sends back an interrupt, informing the network layer it is ready to receive the next packet. The process diagram is shown in Figure 4.4. When a packet is copied to the NIC firmware, it will not be sent out immediately. Before packet transmission is initialized, the MAC layer needs to "capture" the wireless medium following the access rules of the CSMA /CA protocol. It is also possible that when the packet enters the buffer of the NIC firmware, it handles over the packet waiting for service. If the firmware's buffer is not full, the firmware sends a buffer cleared interrupt to the NIC driver and a new packet is copied from the network layer to the firmware buffer. If the entire firmware buffer is occupied, the NIC driver has to wait until a "slot" becomes available (empty) in the firmware buffer. Only after the packet at the head of the firmware's queue is sent out successfully and the ACK message is returned, or the packet at the head of the queue is dropped because all the allowed attempts for retransmission failed, a buffer slot will empty and the firmware will send the buffer cleared interrupt require for a new packet.

If the incoming rate of the traffic steam is below the wireless LAN's NIC transmission speed, the firmware's buffer will have available space to accept the next packet. The packet is sent out immediately after it is copied to the firmware buffer. In this case, the firmware buffer is never full, which means the NIC driver is able to get the buffer cleared interrupt event before the packet is sent out to the Wireless Transmission Medium (the only time needed is the process time for copying the packet from one buffer to another buffer). However, if network

congestion happens, or if the data rate of the traffic stream is higher than the wireless LAN's NIC transmission speed, packets will accumulate in the firmware buffer. Finally, the firmware buffer will fill and a bigger NIC delay will be produced. Equation 4.1 expresses the NIC delay d_{nic}:

$$d_{nic} = \begin{cases} d_{process} & n_{empty} > 0 \\ d_{mac} & n_{empty} = 0 \end{cases} \quad (4.1)$$

Where $d_{process}$ is the process time for copying packet from network layer buffer to firmware buffer by NIC driver; d_{mac} is the delay counting for the time elapsing between the moment a frame is transmitted to the moment the frame is acknowledged. n_{empty} represents the empty slots in the firmware buffer.

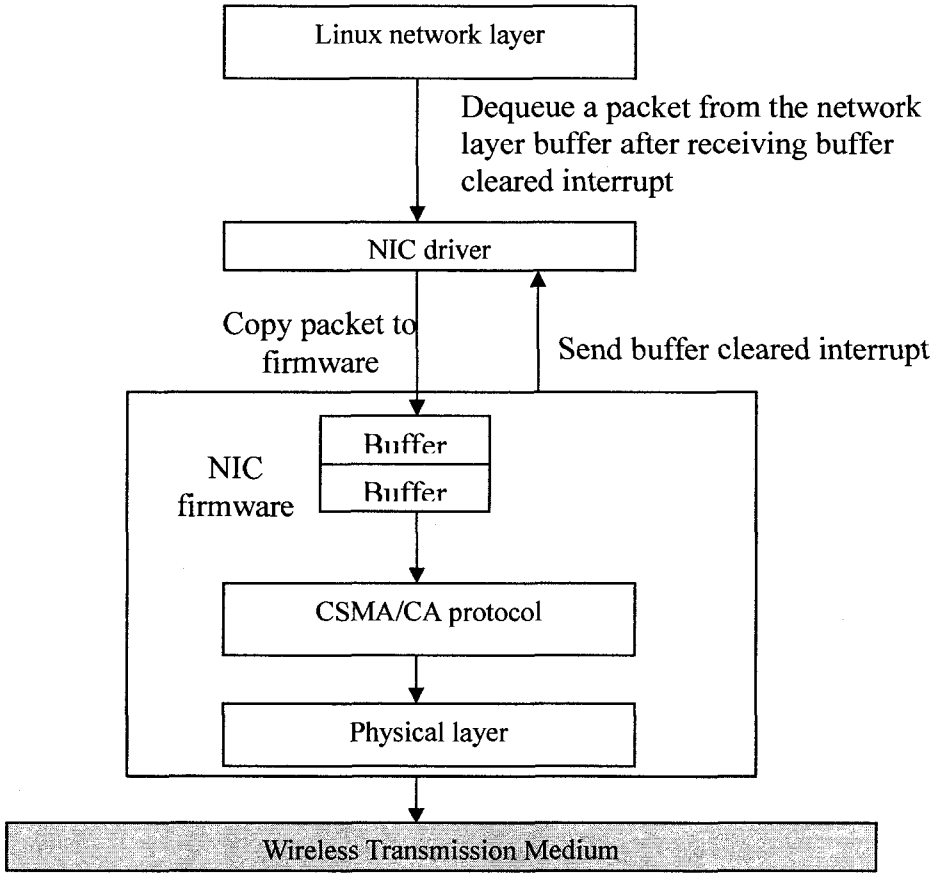


Figure 4.4 Diagram of packet path from Linux network layer to Wireless Transmission Medium

4.2.2.1 Evaluation Experiment of NIC delay

The section introduces the evaluation experiment of NIC delay and presents the test results. In section 5.2.2, the design and implementation of a NIC delay monitor used with the Orinoco wireless LAN network card is discussed. By monitoring the changes of the NIC delay value, the distributed network congestion control scheme is able to determine the current network congestion status. In order to investigate the relationship between NIC delay and the IEEE802.11b wireless LAN NIC transmission capability, a test case is performed, using patched Orinoco IEEE802.11b wireless LAN cards.

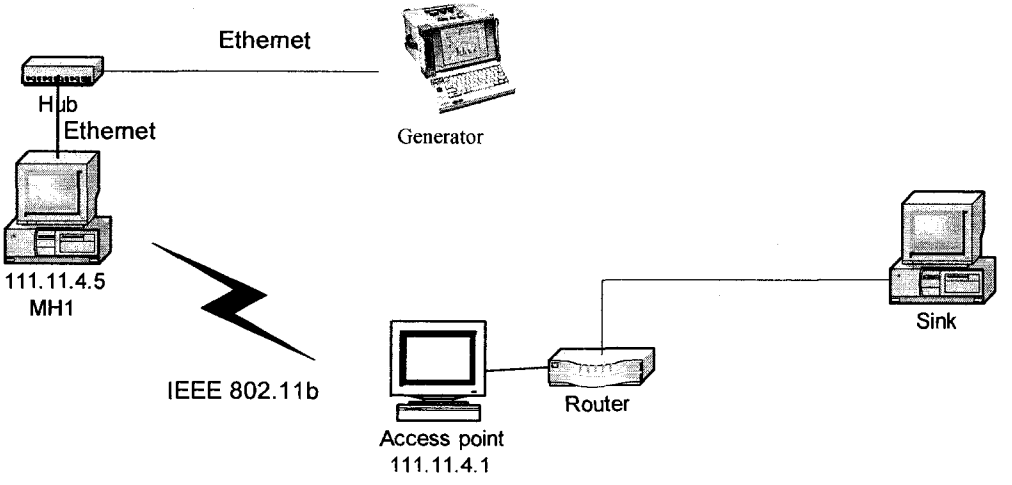


Figure 4.5 Testbed structure for the NIC delay measurement

Test Configuration:

The test configuration is shown in figure 4.5. MH1 is used as sender and AP is used as receiver. The packet payload size is P_l bytes, the buffer size at the network layer is BS^{NL}, the traffic is UDP HIGH-LOW traffic with HIGH period = T_H seconds, LOW period = T_L seconds, the traffic volume during HIGH periods R_H and the traffic volume during the LOW periods is R_L. The networks transmission speed is C_N.

Measured Parameters:

- The NIC delay $d_n(i)$, the ID of the packet associated with the specific $d_n(i)$, the time the packet was delivered to the driver in order to be delivered to the NIC card.

Result Curves:

- The NIC delay $d_n(i)$ versus time.

We collected the test results with different traffic volume during the HIGH period R_H, which are shown in Figure 4.6 to Figure 4.9. Table 4.1 shows the test parameters.

Table 4.1 Parameters for NIC delay evaluation test

Parameters	Value
Packet payload size (P_l)	1000 bytes
Network buffer size (BS^{NL})	5 packets
HIGH period (T_H)	10 seconds
LOW period (T_L)	10 seconds
Traffic volume during the HIGH periods (R_H)	2 Mbps, 3 Mbps, 4 Mbps, 6 Mbps
Traffic volume during the LOW periods (R_L)	1 Mbps
Link transmission speed (C_N)	5.5 Mbps

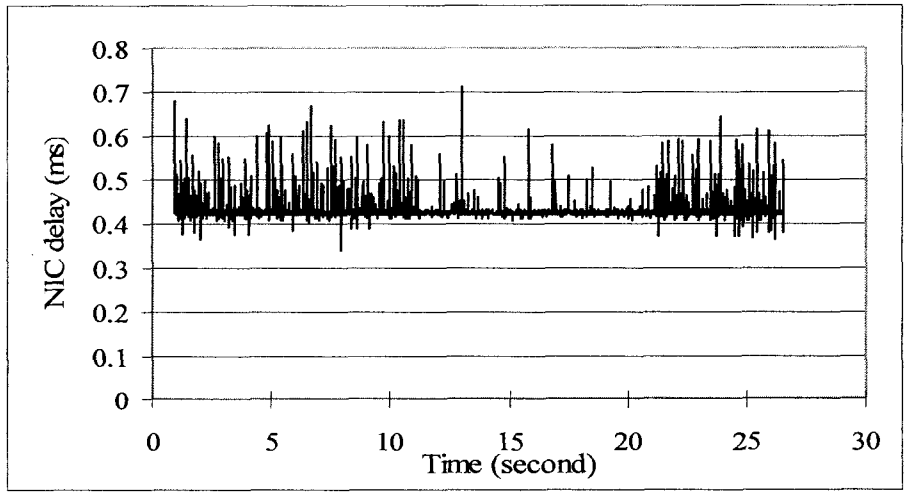


Figure 4.6 The NIC delay when the traffic volume during the HIGH period is 2 Mbps

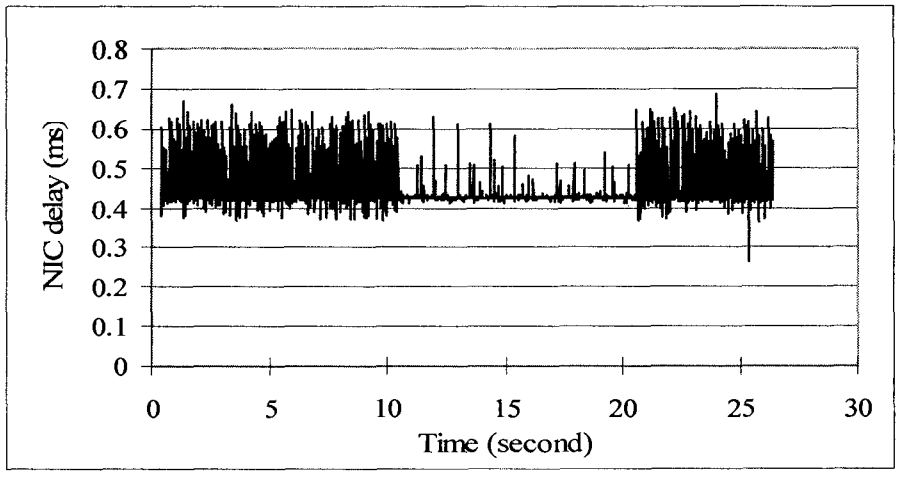


Figure 4.7 The NIC delay when the traffic volume during the HIGH period is 3 Mbps

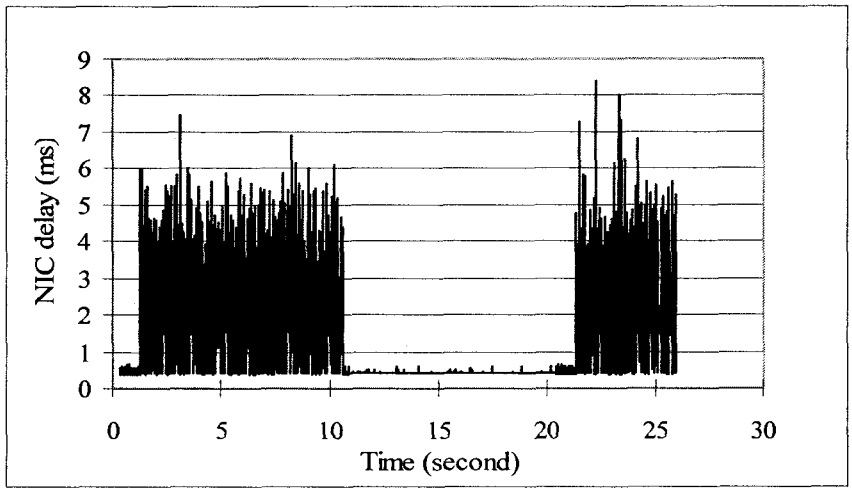


Figure 4.8 The NIC delay when the traffic volume during the HIGH period is 4 Mbps

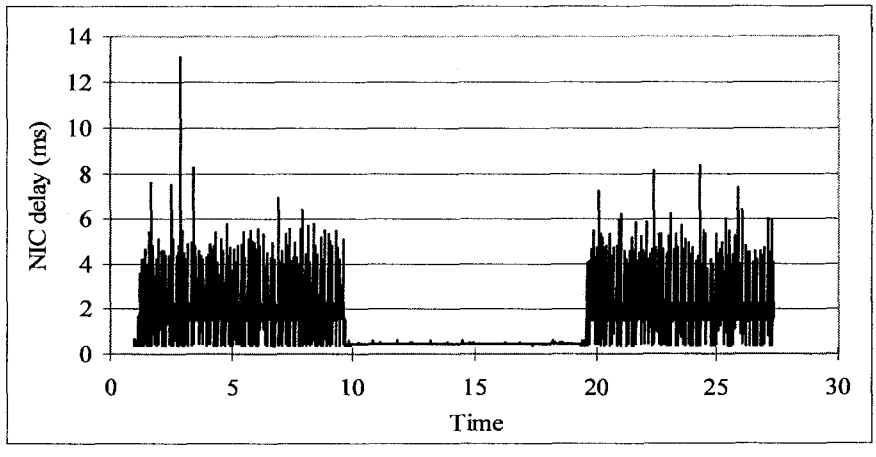


Figure 4.9 The NIC delay when the traffic volume during the HIGH period is 6 Mbps

From the results, we can observe that before the traffic rate reaches the limit of the wireless LAN's effective transmission capacity (3.3 Mbps), the NIC delay is a very small value, less than 1 ms. This time corresponds to the time consumed to copy the packet from the Linux network layer to the Orinoco wireless card buffer. With the traffic increasing, the wireless card transmission ability reaches its limit. The CSMA/CA MAC layer cannot send the packet from the firmware buffer before the next packet is copied into the buffer. As result, the buffer becomes full. The measured NIC delay equals to the MAC delay, i.e. the duration it takes to send a frame out and receive back the acknowledgement indicating the frame was received correctly. For this configuration and speeds, the MAC delay is about 1.6 ms when the transmission data rate is set as 5.5 Mbps.

The Figure 4.10 and Figure 4.11 show the relationship between the NIC delay and the wireless network load while the WALN maximum transmission rate is set to 5.5Mbps and 11Mbps respectively. When the WALN maximum transmission rate is 5.5Mbps(Figure 4.10), the NIC delay has a very small value (less than 0.5ms) before the traffic exceeds the capacity of the network (about 3Mbps). From the previous analysis, we know this small value is not the transmission delay, but the time used in copying a packet from driver to firmware buffer. Only after the traffic data rate exceeds the network capacity, the wireless network becomes overloaded. Now the measured NIC delay equals to the transmission delay (about 1.5ms). From Figure 4.11, we can observe that the NIC delay varies in the same pattern when the traffic is increase from 0 to 8Mbps. The results show that the NIC delays in both tests are same values before congestion happens, which is because the NIC delay measure in that period is the packet copying time form Linux driver to WLAN card firmware. However, after congestions happens, the NIC delay measured is equal to the real MAC layer transmission delay, which is less when the transmission rate is set as 11 Mbps than the value when the transmission rate is set as 5.5Mbps.

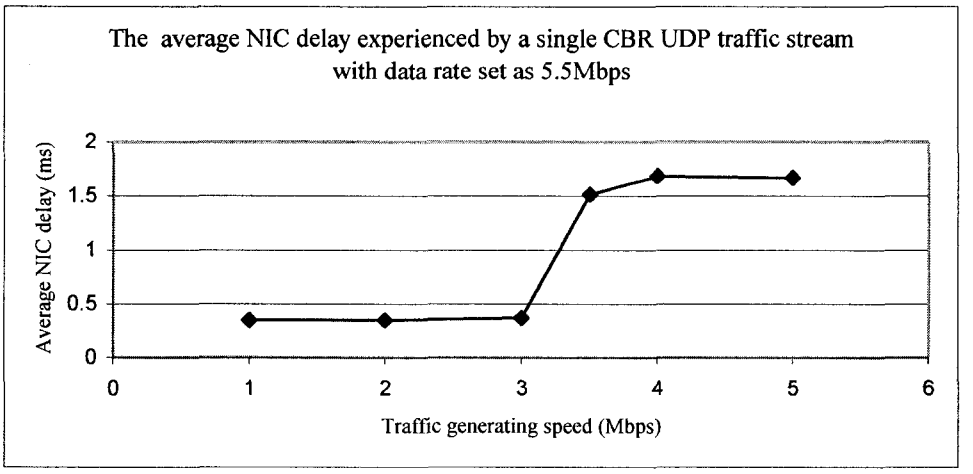


Figure 4.10 The average NIC delay vs traffic volume for a CBR UDP traffic stream when the NIC's transmission rate is 5.5 Mbps

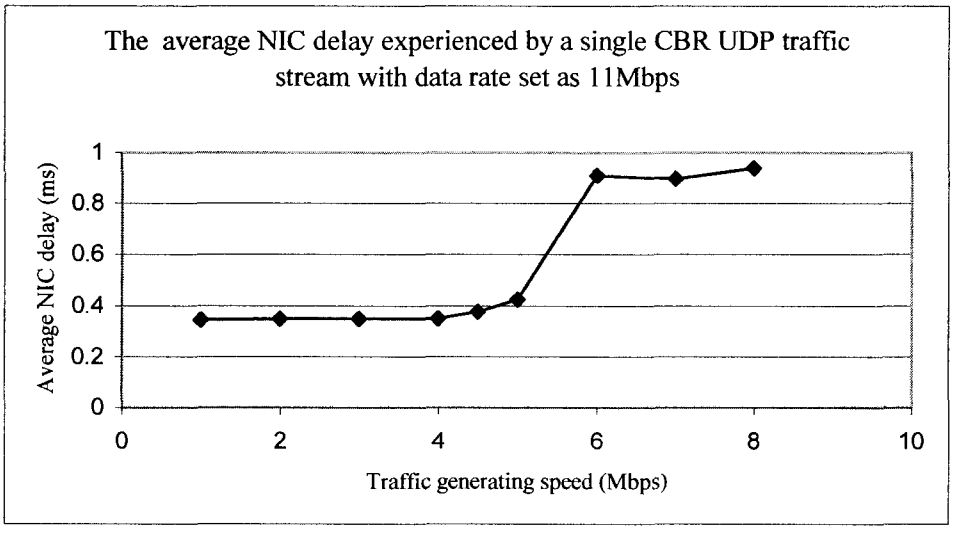


Figure 4.11 The average NIC delay vs traffic volume for a CBR UDP traffic stream when the NIC's transmission rate is 11 Mbps

The results of the NIC delay evaluation tests prove that by monitoring the change of the NIC delay, we are able to identify when the network approaches the point of congestion. This is used from the proposed QoS scheme.

4.2.3 EF service

The characteristics of EF class are associated with low, low delay D_{EF}. Each uplink (from users to the AP) EF class has its own queue in the network layer. All EF traffic streams coming to the downlink (from AP to the users) are placed in the same queue (EF queue) and are processed on a first come first serve basis. The EF packets are time stamped when entering the network layer of the node (AP or user station) and processed by the Queuing Delay Based Drop Module before being sent to the link layer. If a packet remains in the network queue for longer than the parameter D_{buffer}, it is dropped. The parameter D_{buffer} represents the maximum queuing delay allowed for EF packets and is calculated as

$$D_{buffer} = D_{EF} - d_{th}, \tag{4.2}$$

where d_{th} is the delay within the wireless access medium of the system without network congestion.

EF packets entering the AP from the "backbone" network are passed to the EF class without policing, because the edge router of the backbone network has policed the traffic. However, packets produced by the user application are going through policing, applied at the network layer of the user terminals. Each EF application is provided an average rate R_{av} and burst size of M_{burst} packets (transmitted at the maximum rate)[2]. Packets of an application that are violating the "rules" are dropped. Packets of an application that are in "synch" with the "rules" are considered for transmission. The system is supposed to be able to process such packets with maximum dropping rate of (P_{EF}^{L}) and deliver them to the other side with delay not exceeding D_{EF}. If one of the conforming packets is not processed within the maximum delay D_{EF}, it is dropped.

4.2.4 AF/BE service

AF/BE class (C_{AF}^{i}) packets are placed in their corresponding queues. Each queue is processed using a Token Bucket (TB) shaper, which will be described in section 5.1.4. AF/BE class traffic streams are transmitting at a rate $R_{AF}^{i}(t)$ at certain time t (rate the

[2] These two parameters are used in the Token Bucket Queue, which is described in detail in section 5.1.4.

corresponding TB generates tokens). We are allowing a maximum of M^i_{AF} packets to accumulate in a buffer. In order to make sure that the transmission of AF/BE traffics can utilize the available network bandwidth and will not congested the wireless network at the same time, a data rate controller is designed to dynamically adapt the rate $R^i_{AF}(t)$ using the additive increase multiplicative decrease (AIMD) algorithm [CHIU89].

4.2.5 Rate Control

The data rates $R^i_{AF}(t)$ of the traffic shapers are adapted every T milliseconds based on the NIC delay d_{nic}, which is estimated by using the Exponential Moving Average (EMA) algorithm:

$$d_{nic}(i) = a \times d_{nic}(i) + (1-a) \times \hat{d}_{nic}(i-1) \qquad (5.10)$$

where $0 \le a \le 1$, $d_n(i)$ is currently measured value, $\hat{d}_n(i-1)$ is the value calculated in the previous estimation .

Based on the value of the local NIC delay, which reflects the current network congestion status, the new proposed distributed congestion control scheme implements the following rate control rules:

[1] When the NIC delay d_{nic} of the EF traffic exceeds the desired threshold d_{th}, a data rate decrease request message[3] is broadcasted to all stations.

[2] When one station receives the data rate decrease request message or it finds that the local NIC delay d_{nic} exceeds the threshold d_{th}, it decreases the data rate of the low priority traffic (if it has such traffic flows). The system reduces the rate from $R^i_{low}(t)$ to $R^i_{low}(t+T)$ as follows:

$$R^i_{low}(t+T) = R^i_{low}(t) \cdot r^i_{de} \qquad (4.3)$$

[3] The definition of this message will be described in the next Chapter.

where r^i_{de} is the multiplicative decrease factor, which is kept less than 1. The following relations apply between the various parameters.

$$\frac{r^i_{de}}{r^{i-1}_{de}} = R_{step}, \quad R_{step} \leq 1 \ ^4 \tag{4.4}$$

[3] When a station finds the local NIC delay d_{nic} is lower than the set delay threshold d_{th}, it can increase the data rate of the low priority traffics (if it has such traffic flows). The system increases the rate from $R^i_{low}(t)$ to $R^i_{low}(t+T)$ as:

$$R^i_{low}(t+T) = R^i_{low}(t) + c^i_{in} \tag{4.5}$$

where c^i_{in} is the additive increasing data rate factor.

SWAN operates on best effort MAC and uses packet delay feedback-based mechanisms to support soft real-time services and service differentiation in mobile ad hoc networks. The rate control of TCP and UDP best effort traffic is performed at every mobile node in a fully distributed and decentralized manner. In order to achieve the bandwidth and delay requirements for the real time UDP traffic streams, the rate control of SWAN needs to know the MAC layer transmission delay, which is not provided in current wireless network card. In the implementation of the SWAN using commercial available wireless card, the designer used the packet transmission delay, which is measured in network layer, to simulate the MAC transmission delay. However, since the packet delay cannot indicate the real congestion status in MAC layer, SWAN need a longer time to adjust the data rate and resolve the traffic congestion. So the dependency on the MAC delay makes it difficult to use SWAN in the commercial available wireless products.

On the contrary to SWAN, we proposed the distributed network congestion control QoS scheme based on NIC delay, which can be measured in the Linux driver of IEEE 802.11 wireless card without modifying the current hardware.

[4] In our implementation, R_{step} is set as 1.

4.3 Comparison of SWAN and Distributed Network Congestion Control Scheme

SWAN operates on best effort MAC and uses packet delay feedback-based mechanisms to support soft real-time services and service differentiation in mobile ad hoc networks. The rate control of TCP and UDP best effort traffic is performed at every mobile node in a fully distributed and decentralized manner. In order to achieve the bandwidth and delay requirements for the real time UDP traffic streams, the rate control of SWAN needs to know the MAC layer transmission delay, which is not provided in current wireless network card. In the implementation of the SWAN using commercial available wireless card, the designer used the packet transmission delay, which is measured in network layer, to simulate the MAC transmission delay. However, since the packet delay cannot indicate the real congestion status in MAC layer, SWAN need a longer time to adjust the data rate and resolve the traffic congestion. So the dependency on the MAC delay makes it difficult to use SWAN in the commercial available wireless products.

On the contrary to SWAN, we proposed the distributed network congestion control QoS scheme based on NIC delay, which can be measured in the Linux driver of IEEE 802.11 wireless card without modifying the current hardware.

4.4 Summary

In this chapter we introduced the system architectures and the functional description of the distributed network congestion control scheme. This QoS scheme is based on the average value of the NIC delay and uses an active notification method to make the data rate adjustment on the mobile nodes across the wireless network. We also presented the analysis and evaluation results related to the dependency the average value of the NIC delay has from the traffic volume.

Chapter 5

Implementation of QoS Capable Wireless LAN Architecture on Linux

This chapter describes the design and implementation of the QoS capable wireless LAN architecture using regular PCs running on Red Hat Linux and off-the-shelf wireless NICs. Section 5.1 introduces the Linux traffic control framework. Section 5.2 presents the implementation details of the distributed network congestion control scheme.

5.1 Linux Traffic Control

The Linux operating system started by Linus Torvalds in 1991 and evolved by the open source community since then. The source code of Linux is freely available under the GNU License. The network protocol stack of Linux is highly flexible. It gives to the users the ability to configure the protocol architecture within Linux to support virtually every known networking protocol, such as Multiple Protocol Label Switching (MPLS), DiffServ and RSVP, etc.

The Linux traffic control [ALM99], [HUB01] has been in use for several years and became a standard component of the Linux kernel (Version 2.3 and above). Figure 5.1 shows how a data packet is processed, passing through the Linux network stack. When the network interface receives a data packet, it passes the packet to the network layer. First the packet may be classified/tagged and policed (accept if it is within the agreed upon "contract" or drop it if it violates the contract). Then, depending on what its destination is (specified by the IP address in the destination field), the packet is processed "upwards" to the transport layer

(if the destination address is that of the node's) or it is passed to the IP forwarding unit in order to send it to the appropriate next node. In the IP forwarding unit, the IP destination values of the forwarded packets and the packets generated by the local applications are checked to decide the next hop for each packet. When the packet reaches the egress part of the traffic control, it is classified and scheduled. The traffic control decides if the packet should be dropped and when the packet should be sent out.

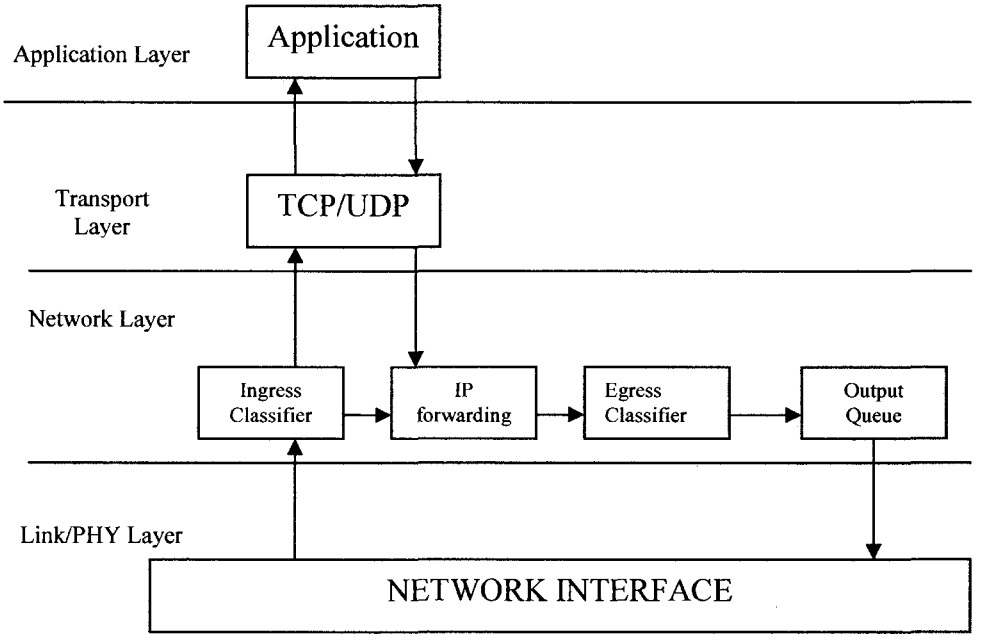


Figure 5.1 Packet processing in the Linux kernel

The traffic control involves the packet processing applied by the ingress classifier/policy, egress classifier/policy and output queuing engines. The basic components of the Linux traffic control includes:

- Queuing Disciplines (Qdisc)
- Classes (within a queuing discipline)
- Filters

Figure 5.2 [ALM99] shows how the basic components work together to perform the traffic control.

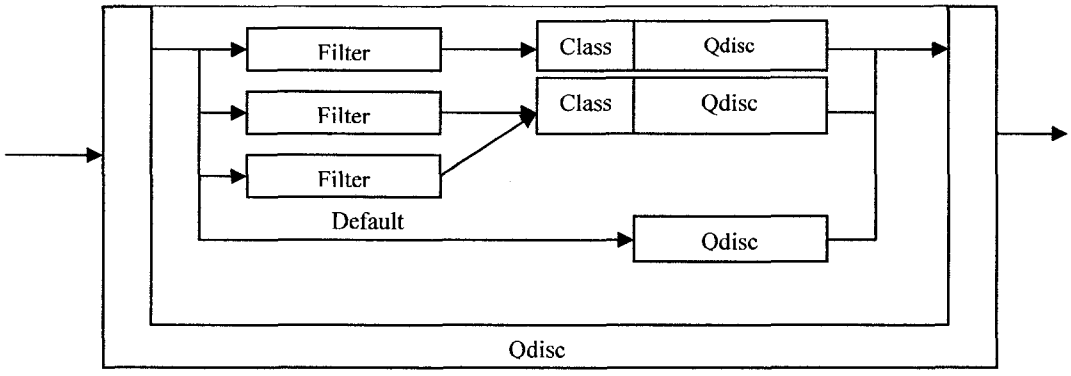


Figure 5.2 Traffic control components

5.1.1 Queuing Disciplines

The queuing discipline (Qdisc) is an algorithm that manipulates the incoming (ingress) or outgoing (egress) queues of a network device. For an enqueued packet, the Qdisc runs one filter after the other until the packet matches a class. Otherwise, the default Qdisc is used. In the case of a match, the packet is enqueued in the Qdisc related to the class for further manipulation. On class is able to be associated with different filters. For each network device, a Qdisc is associated with it. The packets are transmitted from the queue sequentially when the dequeue function is called.

The Qdisc in the Linux kernel can be divided into two categories: classless Qdisc and classful Qdisc.

5.1.1.1 Classless Queuing Disciplines

A classless Qdisc has no configurable internal subdivisions. This category includes the following Qdiscs:

- PFIFO (Priority First In First out)
- SFQ (Stochastic Fair Queuing)
- TBF (Token Bucket Filter)
- RED (Random Early Detection)
- GRED (Generalized RED)

5.1.1.2 Classful Queuing Disciplines

A classful Qdisc contains multiple classes. This category includes the following Qdiscs:

- CBQ (Class Based Queuing)
- HTB (Hierarchical Token Bucket)
- HFSC (Hierarchical Fair Service Curve)

5.1.2 Classes

A classful Qdisc may have several classes, which are internal to the Qdisc. Each of these classes contains an inner Qdisc. The class can be identified in two ways: the class ID, assigned by the user based on the service type of the traffic, or the internal ID, assigned by the queuing discipline. The Class is selected by the enqueue function of the queuing discipline. Then, the enqueue function of the corresponding inner queuing discipline is invoked.

5.1.3 Filters

The filters are used by a queuing discipline for the purpose of assigning incoming packets to one of its classes. Figure 5.3[ALM99] displays the structure of a filter.

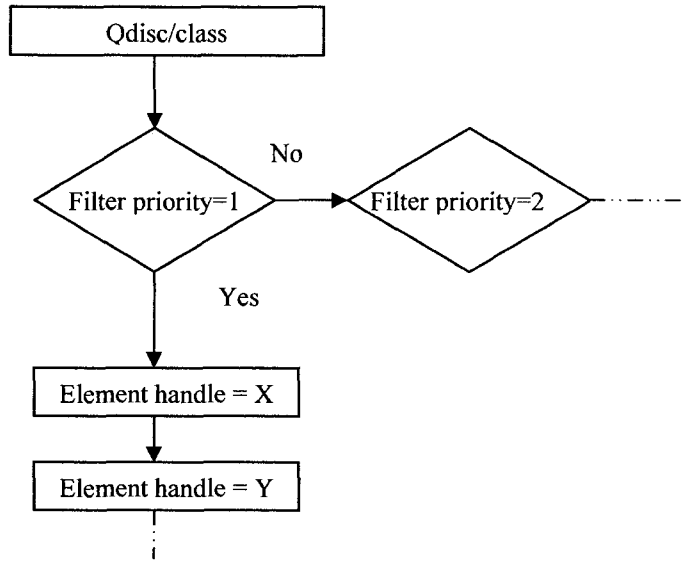


Figure 5.3 Filter structure of traffic control

Filters are kept in filter lists, which are maintained per queuing discipline or per class in ascending priority order.

5.1.4 Hierarchical Token Bucket Traffic queuing discipline

In this section, the hierarchical token bucket (HTB) [MAR02] queuing discipline is presented. In sub-section 5.1.4.1, the implementation of the token bucket algorithm in the Linux kernel is introduced. In sub-section 5.1.4.2, we present an example of the HTB queuing discipline.

5.1.4.1 Implementation of Token Bucket in the Linux Kernel

The token bucket shaper is used to control the bandwidth occupied by a traffic flow by regulating the number of packets entering the network. The token bucket algorithm is shown in figure 5.4.

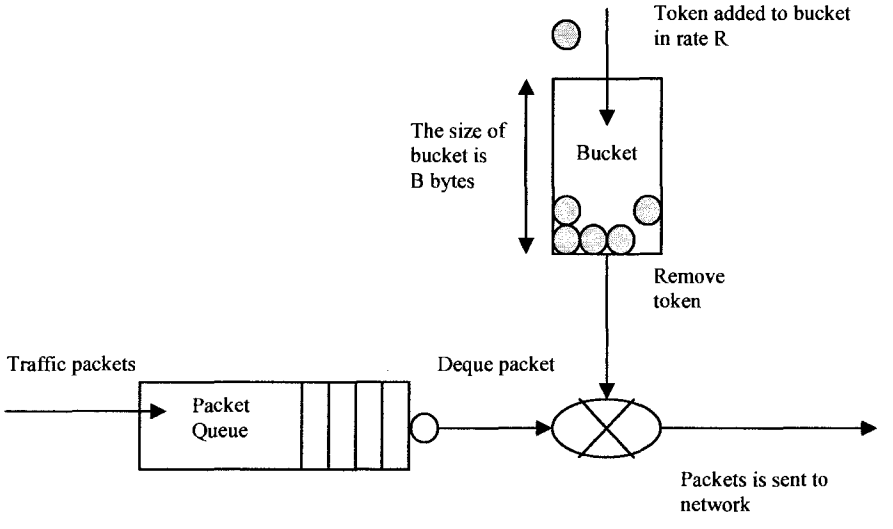


Figure 5.4 Token bucket

A token bucket flow is defined by (R, B), where R represents the rate at which tokens are accumulated and B is the depth of the token pool. For this token bucket flow, in any time interval (t_k, t_f), the number of transmitted bytes M must satisfy the following rule:

$$M \le B + R \cdot (t_f - t_k) \tag{5.1}$$

The HTB Qdisc implements the algorithm by calculating the data rate during the interval time between two transmissions. Assume that packet P_{i-1} was served as time t_{i-1} and assume the remaining tokens in the bucket after serving P_{i-1} are $T_{OK}(t_{i-1})$. The additional tokens accumulated within the interval (t_{i-1}, t_i) are $R \cdot (t_i - t_{i-1})$. The total number of tokens in the bucket right before serving packet P_i is:

$$T_{OK}(t_i) = \min\{R \cdot (t_i - t_{i-1}) + T_{OK}(t_{i-1}), B\}. \tag{5.2}$$

In order to guarantee that the number of tokens will not exceed the upper limit of the token pool, $T_{OK}(t_i)$ is upper bounded by B. If we normalize (5.2) with respect to the rate R, then it becomes:

$$\begin{aligned} \tau_{OK}(t_i) &= \min\{(t_i - t_{i-1}) + \tau_{OK}(t_{i-1}), N\} \\ &= \min\{(t_i - t_{i-1}), N - \tau_{OK}(t_{i-1})\} + \tau_{OK}(t_{i-1}), \end{aligned} \tag{5.3}$$

where

$$\tau_{OK}(t_i) = \frac{T_{OK}(t_i)}{R}, \tag{5.4}$$

$$\tau_{OK}(t_{i-1}) = \frac{T_{OK}(t_{i-1})}{R}, \tag{5.5}$$

$$N = \frac{B}{R}. \tag{5.6}$$

The packet P_i can be transmitted only when there are enough tokens for it. Therefore, the packet P_i can be transmitted only when

$$\tau_{OK}(t_i) \ge S(P_i)/R, \tag{5.7}$$

where $S(P_i)$ is the length of packet P_i. Then, the unused tokens are kept in the pool for the next transmission

$$\tau_{OK}(t_i) = \tau_{OK}(t_{i-1}) - S(P_i)/R. \tag{5.8}$$

If equation (5.7) is not satisfied, the packet P_i remains in the queue, waiting for enough tokens to be accumulated, in order to be serviced.

In HTB Qdisc, two token buckets are applied to one data stream. One of them controls the average rate; the other one controls the peak rate. The goals of HTB Qdisc are as those of the CBQ [FLO95] Qdisc, which includes:

1. Each sub-class should receive roughly its allocated link-sharing bandwidth over appropriate time intervals, given sufficient demand.

2. If all classes with sufficient demand have received at least their allocated link-sharing bandwidth, the distribution of any excess bandwidth should not be arbitrary but should follow some set of reasonable guidelines, which includes:

a. The class is not over limit, OR

b. The class has not-over limit ancestor at level *i*, and there are no unsatisfied classes in the link-sharing structure at level lower than *i*. [FLO95]

5.1.4.2 Example of the HTB Queuing Discipline

We consider a typical case of traffic control applied in a small business environment, which is using a Linux based router, applying traffic control. Two sub-networks share one 100 kbps line to access the Internet. The network is illustrated in figure 5.5. We want to allocate 40 kbps to subnet A (192.168.1.0) for employees to surf the Internet. In order to make the WWW server, which belongs the subnet B (192.168.2.0), work properly, we reserve for it 60 kbps bandwidth for it. At times, when less bandwidth is needed by one subnet, the unused bandwidth will become available to the users of the other subnet.

To realize the above bandwidth allocation, the HTB traffic queuing discipline is deployed on eth0 of the edge router, controlling the bandwidth allocation of the outgoing traffic. The configuration applied to eth0 is shown in figure 5.6. The class and U32 filter components are used in this configuration, with the setup steps being as follows:

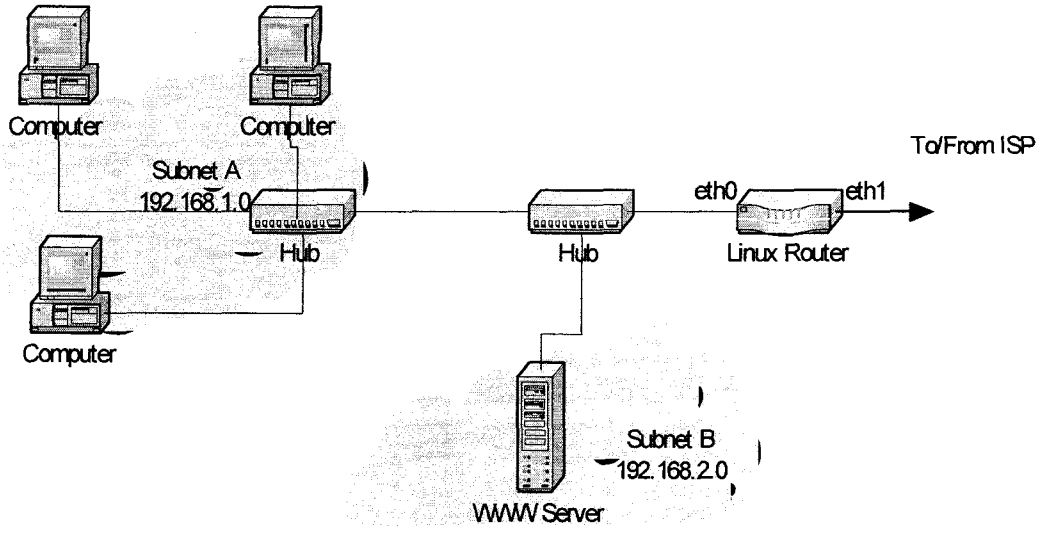


Figure 5.5 A representative office and small business network, using a Linux based router, running HTB

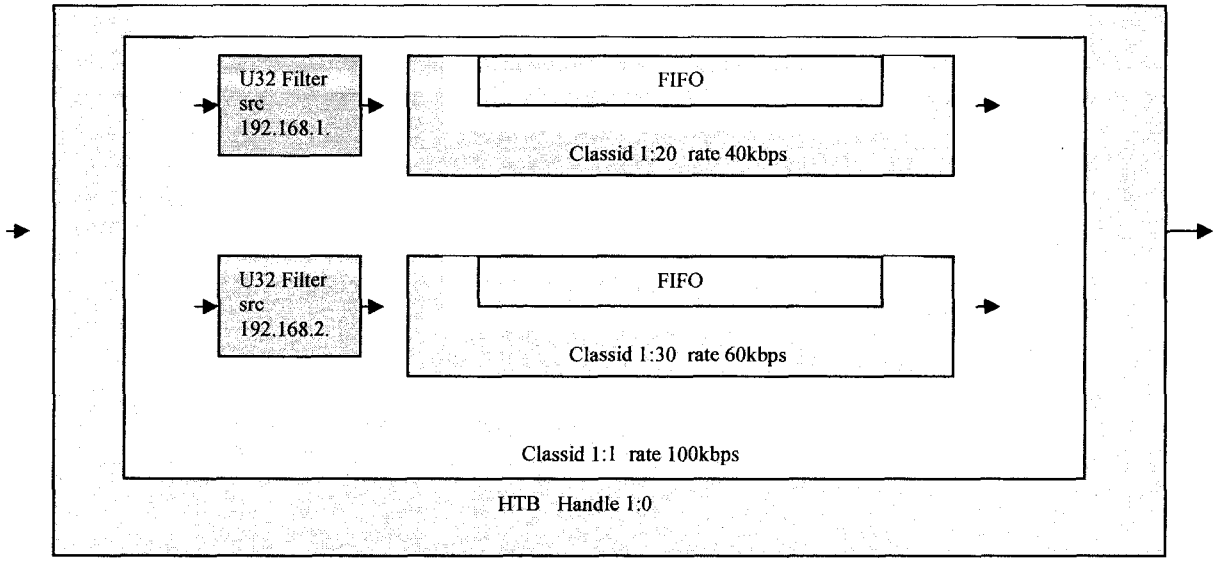


Figure 5.6 Configuration of NIC eth0 on Linux router

(1) At first reset the Qdisc of the NIC eth0:

tc Qdisc del dev eth0 root

(2) Then a root Qdisc is added which uses the class-based queuing scheduler:

tc Qdisc add dev eth0 root handle 1: htb default 12

(3) The next step is to create a root class which contains two sub-classes. One is for the WWW traffic, the other is for the traffic coming from the workstations:

tc class add dev eth0 parent 1:0 classid 1:1 htb rate 100kbit ceil 100kbit

tc class add dev eth0 parent 1:1 classid 1:20 htb rate 40kbit ceil 100kbit

tc class add dev eth0 parent 1:1 classid 1:30 htb rate 60kbit ceil 100kbit

(4) The final step is to bind two U32 filters to the root Qdisc. The filters decide to which class a packet belongs based on the source IP address of the IP packet:

tc filter add dev eth0 parent 1:0 protocol ip prio 4 handle 1: u32 divisor 1

tc filter add dev eth0 parent 1:0 protocol ip prio 5 handle 1: u32 divisor 1

tc filter add dev eth0 parent 1:0 prio 4 u32 match ip src 192.168.1.0/24 flowid 1:20

tc filter add dev eth0 parent 1:0 prio 5 u32 match ip src 192.168.2.1 flowid 1:3

5.2 Implementation of the Distributed Network Congestion Control Scheme

This section presents the design and implementation of the distributed network congestion control scheme on Red Hat Linux. This scheme is developed in the kernel space of Red Hat 7.3 Linux using the kernel version 2.4.17.

5.2.1 Queuing Delay Based Drop module

The task of the queuing delay based drop module is to discard the EF traffic packets which have been waiting in the network layer buffer for too long. The functionalities of this module were introduced in section 4.2.3. Two functions are patched into the Linux kernel source code. One function adds time-stamps to the EF traffic packets when the packets enter the network layer. The other reads the time-stamp and determines whether to drop or send the packet.

5.2.1.1 Add time-stamp

The Linux network stack is based on the TCP/IP protocol architecture. Figure 5.7 shows how the Linux network stack works.

The user application uses TCP/UDP socket to transmit packets. Data packets are sent to transport layer (TCP or UDP) through sockets and then are copied to the network layer (IP). In the IP layer, the kernel looks up the route based on the destination information stored in the packet header. If the destination is another host, the kernel addresses it and then sends the packet to the corresponding network device, which ultimately sends the packet out over the physical medium.

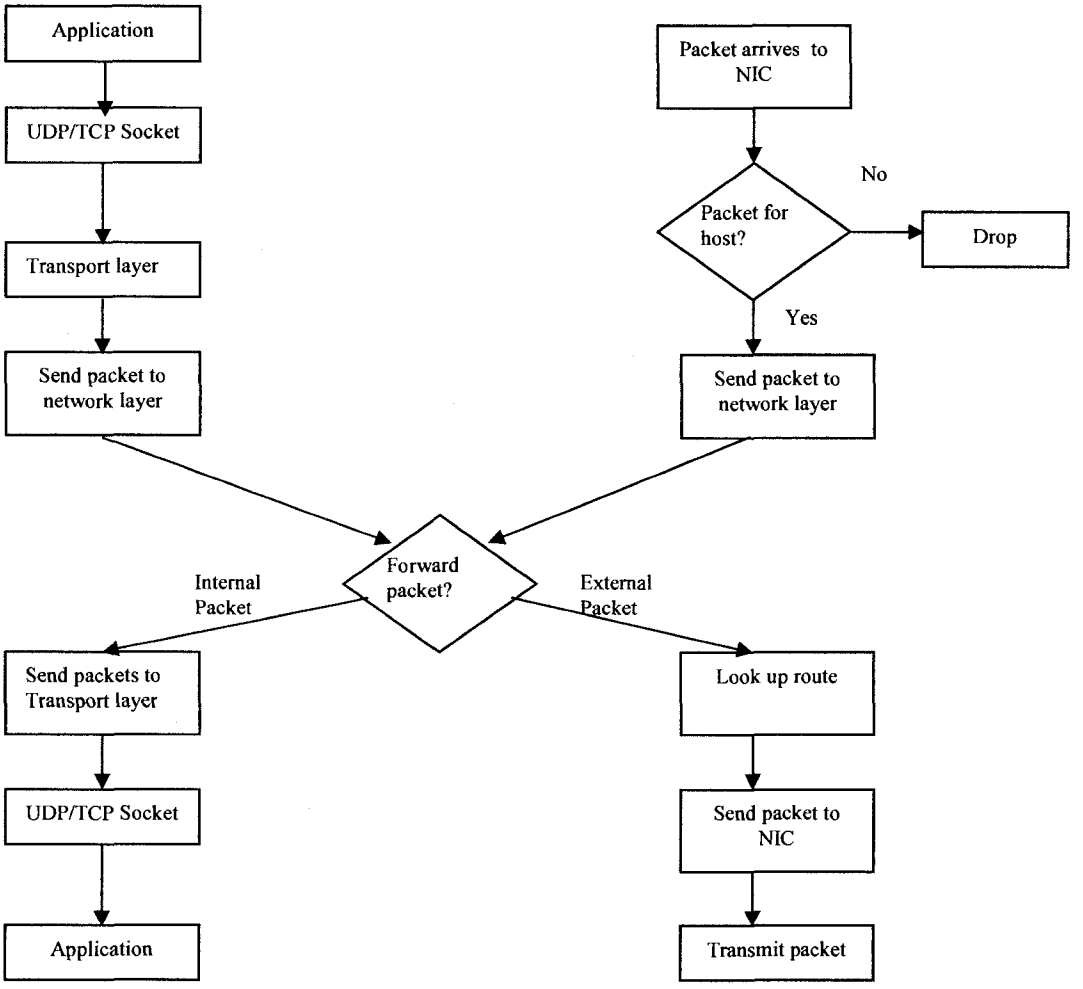


Figure 5.7 Traffic paths in the Linux network stack

On the other hand, when a packet arrives at the network interface card (NIC) from the network, the NIC checks the MAC address which is stored in the packet header to decide if the packet is for the host computer. If so, the packet is sent upwards to the IP layer, where the kernel looks up the destination address contained in the IP header of the packet. If the packet has to be forwarded to another computer, the IP layer sends it back downward to an output interface. If the host computer is identified as the receiver of the packet, it is sent upwards, passing through the transport layer and sockets, finally reaching its application.

In the Linux network stack source code, the common packet data sk_buff (defined in /include/linux/skbuff.h) is used to carry user data and network protocol header. The structure of sk_buff is shown in table 5.1.

Table 5.1 Structures of sk_buff

Type	Name	comment
struct sk_buff *	next	Next buffer in sk_buff list
struct sk_buff *	prev	previous buffer in sk_buff list
struct sk_buff_head*	list	The sk_buff list
struct sock *	sk	The socket this sk_buff belongs to
struct timeval	stamp	Struct to record arrive time
struct net_device	dev	Device the sk_buff used
Union	h	Transport layer header
Union	nh	Network layer header
Union	mac	Link layer header
struct dst_entry	dst	Destination of the sk_buff
char	cb[48]	Control buffer
Int	len	The length of the data
Int	csum	Checksum
Char	pkt_type	Packet class
_u32	priority	Packet queuing priority
Unsigned short	protocol	Packet protocol from driver
Unsigned short	security	Packet security level
Unsigned char *	head	Pointer to buffer header
Unsigned char *	data	Pointer to data head
Unsigned char *	tail	Pointer to buffer tail
Unsigned char *	end	Point to end
Void	(*destructor)(sk_buff *)	Pointer to destruct function

In our implementation of the queuing delay based drop module, we utilize the stamp field of the sk_buff structure to add a time-stamp, which records the time when an outgoing packet enters the network layer from the transportation layer. In the original Linux network stack functions, this field is used by the NIC driver to record the time when the packet arrives at the host and is not used (which is set as zero) when the sk_buff carries an outgoing packet. We can use this stamp field without affecting the current Linux network stack functions.

When a packet goes down to the network layer, a sk_buff structure is created in function **ip_build_xmit** [5] to contain the user data and the transportation layer header. At the same time, the IP header is attached to this structure. In order to add the time-stamp, we modify the function **ip_build_xmit** and add **get_fast_time** function to save the time in the stamp field of the sck_buff structure.

5.2.1.2 Drop an EF Packet According to the Time_stamp

After the packet is processed at the network layer, it is sent into the outgoing queue of the network device, waiting to be dequeued to the device driver. However, if the network is congested because of heavy traffic load, the packet has to stay in the queue for a long time. In order to maintain the delay bound of the EF traffic, the delayed packets must be dropped according to the drop guideline and process, described in section 4.2.3. We patched the drop process code into the Orinoco IEEE802.11b wireless LAN driver function **dldwd_xmit**[6]. Figure 5.8 shows the logical flow of the patched drop module function.

When the modified Linux Orinoco wireless card driver gets the packet (In fact, the driver code gets a pointer pointing to the sk_buff structure which contains the data load and protocol layer headers), the drop module determines if the packet belongs to the EF traffic by

[5]Function ip_build_xmit is located in /net/ipv4/ip_output.c and is called by transportation layer functions when the data is copied from the socket to the IP network layer.
[6]Function dldwd_xmit is Located in the file /drivers/net/wireless/Orinoco.c.

checking the value of Differentiated Service Code Point (DSCP) field in the IP header. For the EF service, the codepoint is 101110 (46).

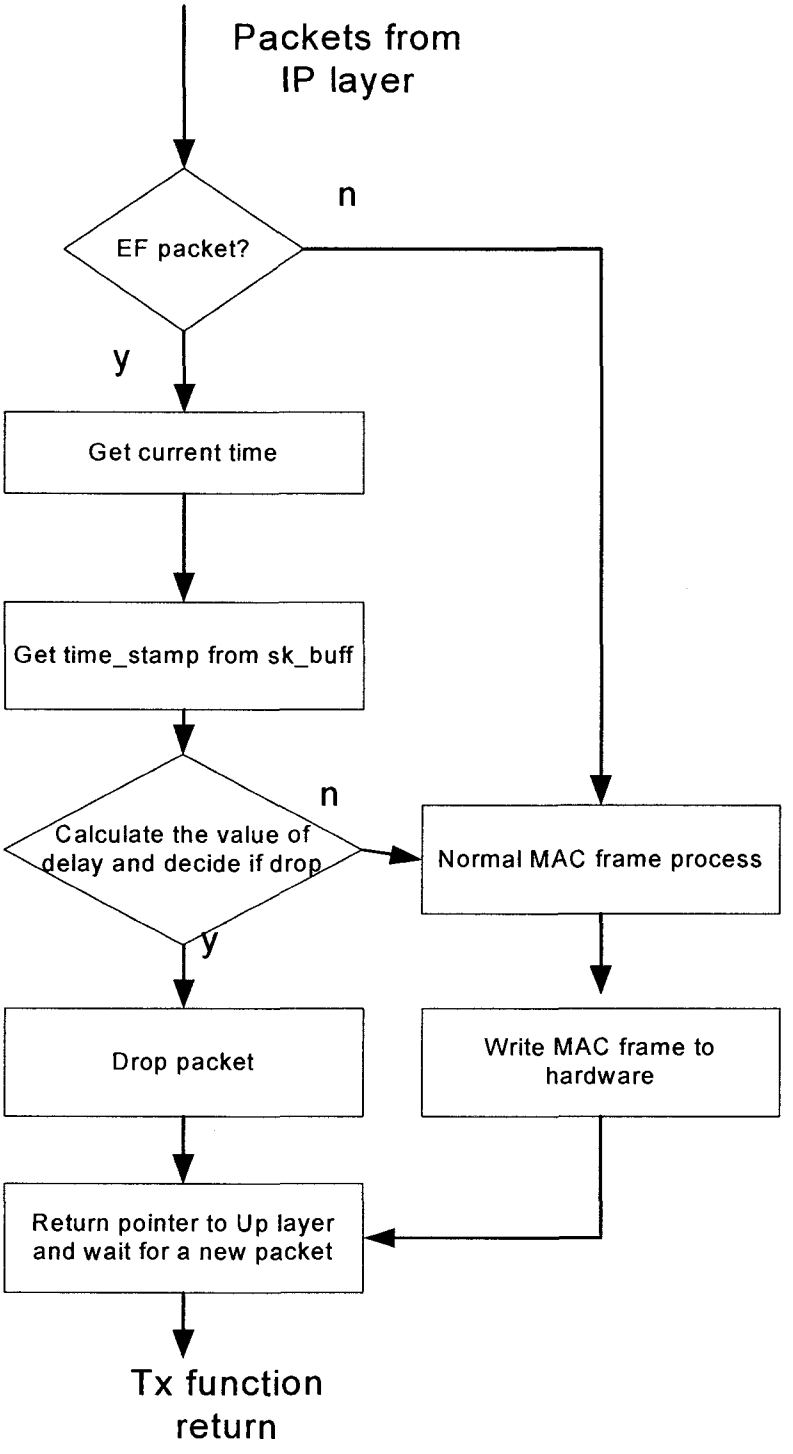


Figure 5.8 Logical flow of the dldwd_xmit function with packet drop module patched

If the packet belongs to the EF traffic, the module gets the current time by calling the **get_fast_time** function and puts the value in a **timeval** structure (defined in /linux/time.h) which is defined as:

```
struct timeval {
    int tv_sec ;    /* seconds */
    int tv_usec ;  /*  microseconds */
};
```

This time is compared with the timestamp in the sk_buff to determine the queuing delay. If the queuing delay exceeds the set maximum allowed delay, the packet is dropped. If not, the MAC header is added to the packet and the driver copies the completed MAC frame to the NIC hardware.

5.2.2 Measurement of the NIC Delay

The packet transmission delay is measured by calculating the buffer clearing time of the Orinoco wireless LAN Network Interface card. In this thesis, we define this buffer clearing time as the NIC delay, which was introduced in section 4.2.2. When network congestion occurs, the NIC has to "back off" and uses more time to transmit the packet. As a result, the packet is kept in the buffer for a longer time, which results to a bigger NIC delay value.

Linux provides several methods to count the time in the kernel space. The timer interrupt mechanism of Linux is used commonly in the device drivers. The Linux kernel uses this timer interrupt mechanism and a special variable named *"jiffies"* to keep track of time [RUB01]. The timer interrupts are generated by the system's timing hardware at regular intervals. This interval time is set by the kernel according to the value of symbol HZ, which is an architecture-dependent value defined in <linux/param.h>. The value of HZ specifies the frequency of the timer interrupts. Current Linux versions define HZ to be 100 as the default value for most platforms, which means that the timer interrupts happen 100 times every second. Every time a timer interrupt occurs, the value of the variable "jiffies" is incremented by one, which represents 10 milliseconds.

However, the accuracy of this mechanism is not good enough for the measurement of the NIC delay in IEEE802.11 wireless LAN, for which the delay value is about one millisecond. Fortunately, Linux provides another platform dependent resource for considerably higher time precision. A register named TSC (timestamp counter) is included in the processor, which counts the CPU clock cycles. The value of the register can be read using the function **rdtsc** or **rdtscl** (included in header file /asm/msr.h). In our testbed, the main frequency of the CPU used in the PC is 1.8 GHz, which means that the time precision we can get is:

$$1/1.8\times10^{9}\ \text{second} = 0.56\times10^{-9}\ \text{second} = 0.56\ \text{nanosecond} \tag{5.9}$$

The estimate of NIC delay $\hat{d}_n$ is calculated by using the Exponential Moving Average (EMA) algorithm:

$$\hat{d}_n(i) = a\times d_n(i) + (1-a)\times \hat{d}_n(i-1) \tag{5.10}$$

where $0 \le a \le 1$, $d_n(i)$ is currently measured value, $\hat{d}_n(i-1)$ is the value calculated in the previous estimation .

5.2.3 Extension of HTB Qdisc interface

This section presents the extensions made to the existing HTB Qdisc in order to enable the nodes to support the implementation of the distributed network congestion control scheme on the IEEE802.11 wireless LAN. The main objectives for the design are:

- Design a mechanism to re-use existing functions/interfaces whenever possible.
- Integration in the current Linux traffic control framework.
- The scheduling parameters (such as average date rate, maximum date rate, etc) can be changed by function calling from other kernel processors (eg. rate controller).

The interface of a queuing discipline is defined in a structure `Qdisc_ops` (This structure specifies the Qdisc operations and is defined in /include/net/pkt_sched.h), whose members are listed in Table 5.2.

Table 5.2 Definition of the structure `Qdisc_ops`

Name	**Description**
Next	The pointer to the next Qdisc in the list
cl_ops	The pointer to the class operation, which is deployed by this Qdisc
Id	The unique name of this kind of Qdisc, such as CBQ, TBF,etc.
priv_size	The size of the private data buffer
enqueue (*skb, *Qdisc)	The enqueue function
dequeue (*Qdisc)	The dequeue function
requeue (*skb, *Qdisc)	Place the skb at the head of the queue if it can't be sent out
drop (*Qdisc)	Drops a packet from the queue
init(*Qdisc, *arg)	Initializes the Qdisc
reset(*Qdisc)	Resets the Qdisc
destroy(*Qdisc)	Deletes the Qdisc and clears the memory
change(*Qdisc, *arg)	Changes the Qdisc parameter
dump(*Qdisc, *skb)	Returns the Qdisc statistics

The HTB Qdisc defines its Qdisc operation struct **htb_Qdisc_ops** in file **sch_htb.c** (located in /net/sched/). The HTB Qdisc operation structure is defined as:

```
struct Qdisc_ops  htb_Qdisc_ops
{
        NULL,
        &htb_class_ops,
        "htb",
        sizeof(struct htb_sched),

        htb_enqueue,
```

```
    htb_dequeue,
    htb_requeue,
    htb_drop,

    htb_init,
    htb_reset,
    htb_destroy,
    NULL /* htb_change */,

    htb_dump,
};
```

From the above definition, we can find that in the original HTB source code, the interface htb_change is defined as a NULL pointer. This means that this interface is not defined and cannot be used. In order to change the Token Bucket parameters (such as assured rate, ceil rate[7], etc,) by function calling from the rate controller in the Linux kernel, we define a new htb_change interface that points to a new function, "htb_test". We patched this function in the file /net/sched/sch_htb.c. The logic flow of this function is shown in figure 5.9.

The first action of the htb_test function is to get the target traffic class from the class list of the HTB Qdisc. Then it reads the control parameters that are encapsulated in the interface calling arguments. Before the token bucket parameters are changed, the parameter "buff" is locked, in order to prevent the parameters from being changed by two different kernel processes running at the same time. The data rate of the token bucket is calculated by using the AIMD algorithm, which was described in section 4.5.4. After the data rate table is updated, the HTB parameter tree is unlocked.

[7] The ceil rate in HTB is the maximum rate one class can get if the bandwidth resource is sufficient.

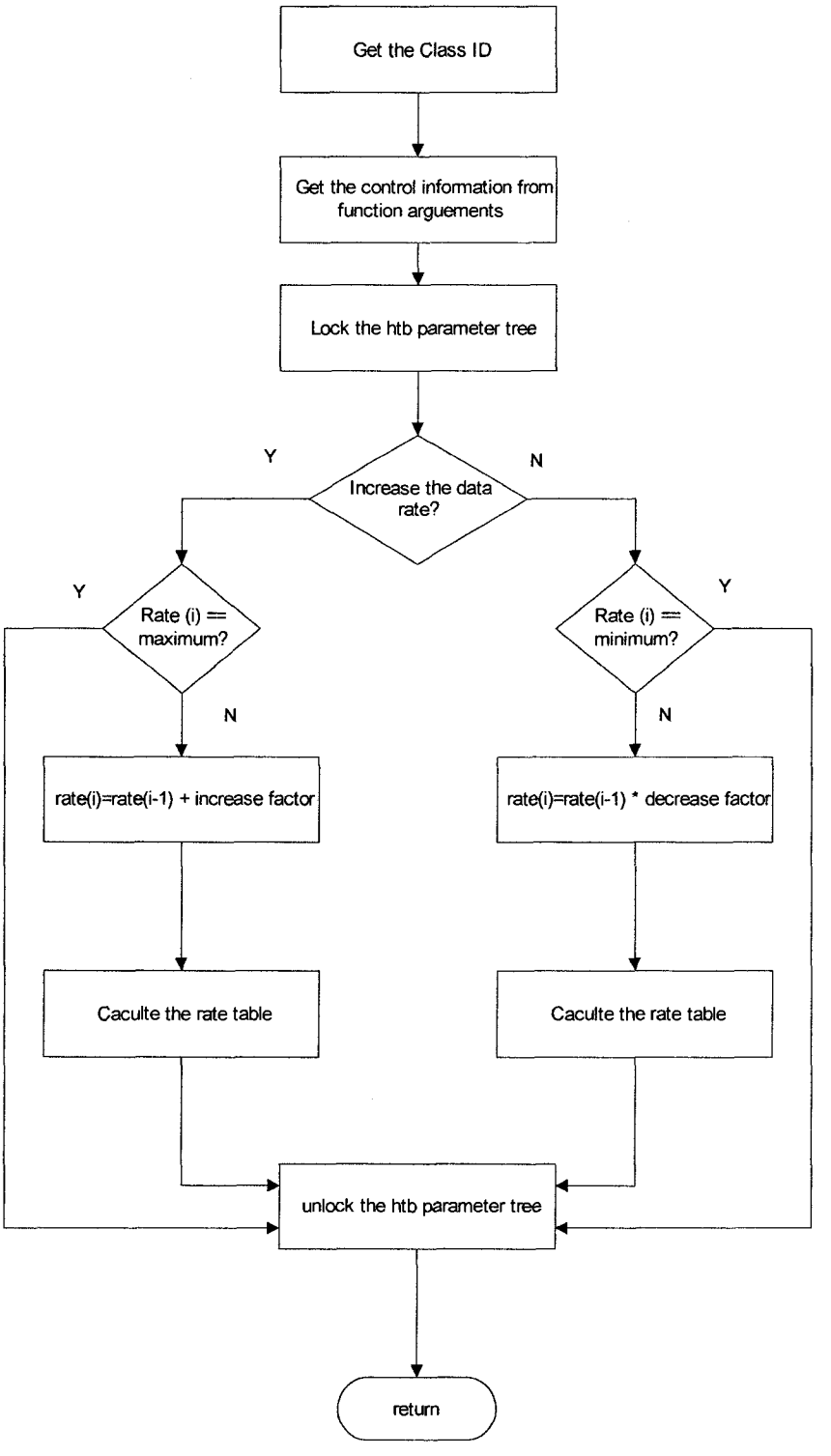


Figure 5.9 Logic flow of the new htb_test change interface function.

5.2.4 Rate Controller

The data rate controller performs the following functions:

1. Broadcasting of the data rate decrease request messages when the NIC delay $\hat{d}_n$ exceeds the set delay threshold d_{th}.
2. Calling the rate-changing interface of the local HTB Qdisc when the station receives a data rate decrease request message, broadcasted by another host.
3. Calling the rate changing interface of the local HTB Qdisc based on the measured NIC delay $\hat{d}_n$

5.2.4.1 Broadcasting data rate decrease request messages

This function module generates the data rate decrease request messages and broadcasts the messages when the measured NIC delay $\hat{d}_n$ exceeds the desired delay d_{th}.

Instead of using UDP/TCP socket, we use the raw socket [STE98] as the broadcast interface. The Raw socket allows the applications to read and write an IPv4 packet with a self-defined IPv4 protocol field that does not belong to the Linux network stack (The Linux network stack recognizes only the packets containing protocol field values of 1 (ICMP), 2 (IGMP), 6(TCP) and 17 (UDP)). A new protocol, the Rate Adaptation Request Protocol (RARP), is defined in this QoS scheme. The field value for the new RARP is set as 100, which is a reserved value of the Linux network stack.

In this function module, the parameter T_update is defined, representing how long is the interval of the NIC delay sample time. The smaller T_update is, the faster the NIC delay is sampled. With the more frequent sampling of the NIC delay, when network congestion occurs, it can be noticed earlier and resolved more quickly. However, if the T_update value is too small, it will increase the workload of the system, as it will consume more processing resource. At the end, this will backfire, generating performance degradation, since the curcial functions will not have enough time to complete.

5.2.4.2 Changing Rate after Receiving RARP Messages

The module implementing this function is patched into the RX function **__dldwd_rx** of the Orinoco wireless LAN NIC driver (Located in file /drivers/net/wireless/Orinoco.c). The logic flow of the patched RX function is shown in figure 5.10.

When the RX function receives a packet captured by the NIC, the IP header of the packet is checked. If the protocol field is 100, the packet is interpreted as a RARP message, broadcasted by another station. The rate controller performs the data rate adaptation. If not, the Linux network stack processes the packet.

When network congestion occurs, stations might broadcast RARP messages simultaneously or close in time. If the controller responds to all of these messages, the local data rate would be reduced repeatedly, driving the AF/BE traffic to starvation. In order to avoid this, the controller checks the arriving time. If several RARP messages arrived within a certain window of time, only the first message is processed, while the others are discarded. Two parameters, L_w and T_{begin}, are defined and used to implement this function. L_w is defined as the length of the valid window time. T_{begin} is initialized as 0 at first. When the first RARP packet arrives, the current time is set to T_{begin}. The arriving time of the following RARP packets will be compared with the T_{begin}. The packets which arrived during the period (T_{begin}, $T_{begin} + L_w$) are considered as declaring the same network congestion status and are discarded. The first packet that arrived out of this range is a valid message and the value of T_{begin} is reset. The diagram describing this function is shown in figure 5.11.

After the changing mode is set as decrease, the htb_change interface of the HTB Qdisc is called through the function pointer **dev->Qdisc->ops->change**[8].

[8] dev is a netdev structure defined in /linux/netdevice.h and qdsic is defined in /net/pkt_sched.h.

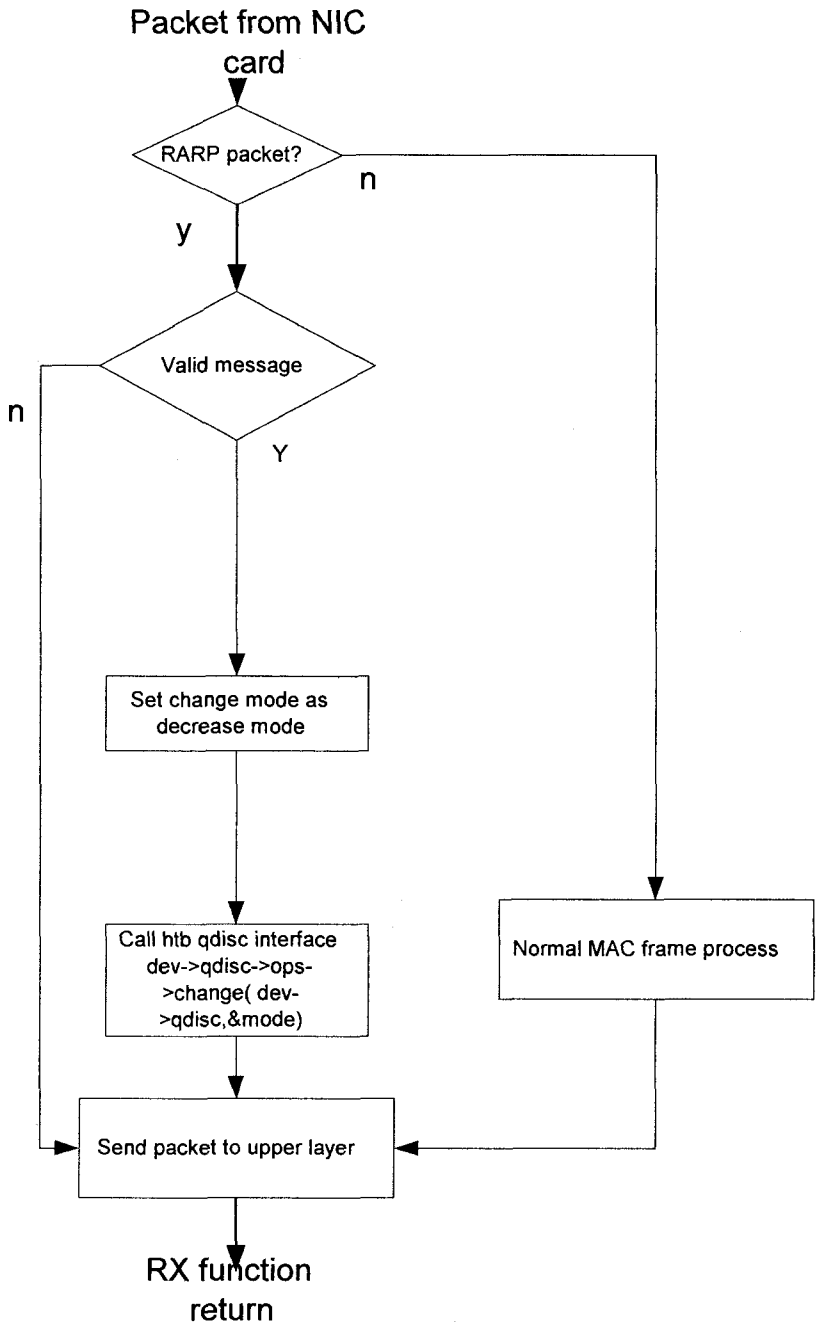


Figure 5.10 Logic flow of the patched RX function

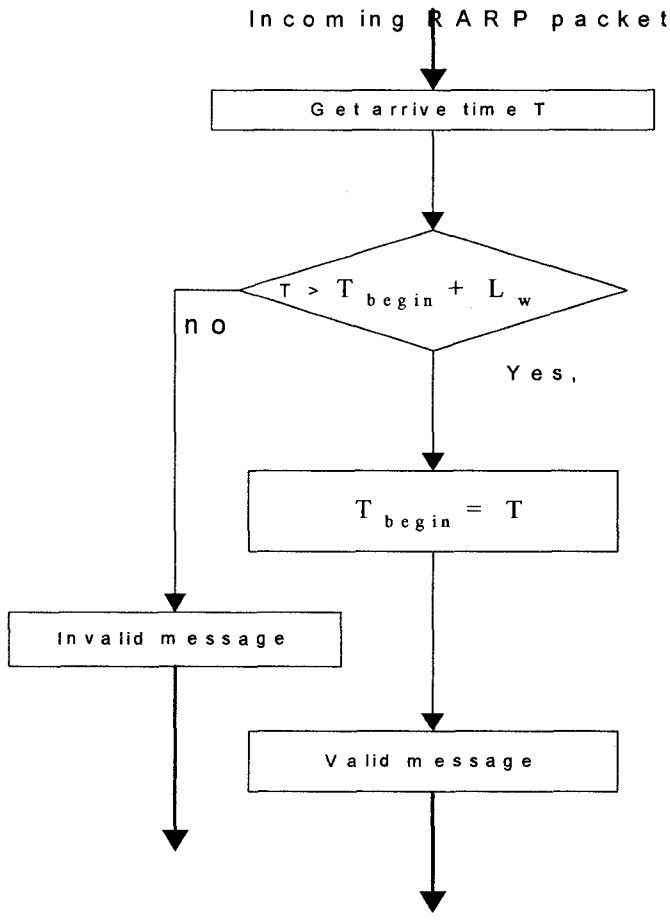


Figure 5.11 The flow diagram describing the process for choosing valid RARP messages

5.2.4.3 Changing Rate based on Measured NIC Delay

In section 5.2.4.1, we defined the update interval T_update. In this function module, a parameter T_inter is defined. Combined with the T_update, T_inter is used to decide if the update time of data rate configuration has arrived. After the system starts, the T_inter is set as 0. Then, its value is increased by 1 at every click of the Linux internal timer. When the value of T_inter reaches the value of T_update, the rate control function is performed and the T_inter is reset to 0. The logic flow of this function module is shown in figure 5.12.

Based on the NIC delay d_n, the rate controller decides whether to increase (the value of the variable *mode* is set as 1) or decrease (the value of variable *mode* is set as 0) the data rate of the AF/BE token bucket queue. The threshold value d_{th} is the desired delay we want to maintain in the wireless network. After the changing mode is decided, the htb_change

interface of the HTB Qdisc is called through the function pointer **dev->Qdisc->ops->change**.

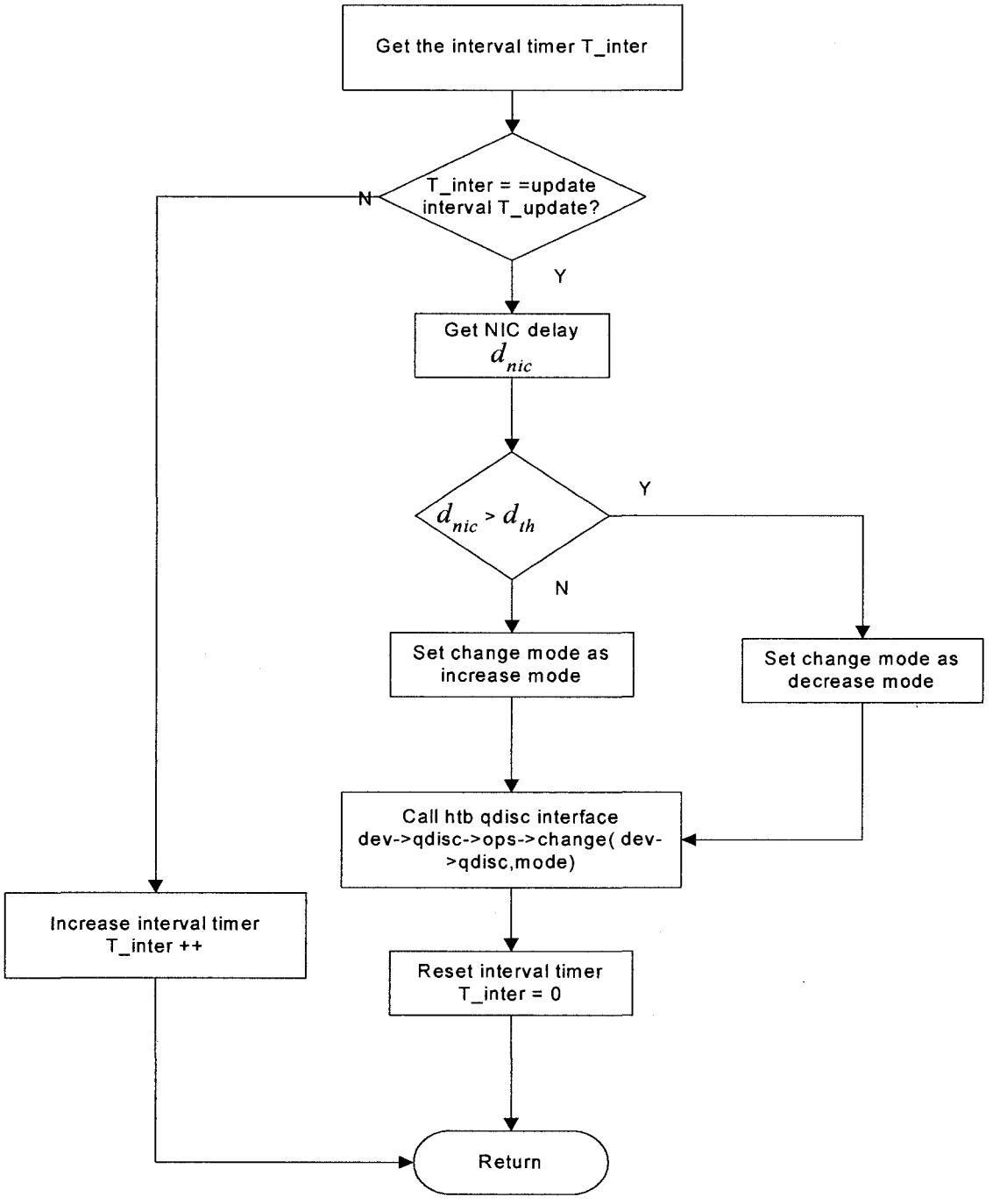


Figure 5.12 Flow diagram of the rate controller function module .

5.3 Summary

In this chapter, we introduced how the distributed network congestion control scheme is implemented on RadHat Linux 7.3 using kernel version 2.4.17. The Linux traffic control framework is presented. One of the Linux hierarchical token buckets, the Hierarchical Token Bucket (HTB) is described in detail using a typical application example. Based on the Linux traffic control framework, the prototype of the new QoS scheme is designed and developed in the Linux kernel space. We patched the distributed network congestion control function into the Orinoco wireless LAN network card driver. The source code of the HTB hierarchical token bucket is also modified, in order to provide an in-kernel function-calling interface. With this interface, the data rate of HTB can be modified by the function of distributed network congestion control, which is much faster than the configuration using tc script.

Chapter 6

IEEE802.11b Wireless LAN Performance Measurements and Analysis

In this chapter, the measurements and performance analysis results are presented. In section 6.1, the hardware and software tools used in our IEEE 802.11b wireless LAN testbed are introduced. Section 6.2 presents a thorough performance evaluation test of Orinoco/Lycsys IEEE 802.11b wireless LAN products.

6.1 Experiment Testbed Setup

This section describes the necessary hardware equipment and software needed to perform the test and the experiments. The hardware is commercially available as described in the section 6.1.1 and the corresponding software is written or modified by the author of this thesis.

6.1.1 Hardware

The testbed includes the following nodes:

Access Point: IBM P4 PC running on the Red Hat Linux 7.3 operation system[9]
Lycsys IEEE802.11b PCMCIA wireless NIC

Station: IBM P4 PC with Red Hat Linux 7.3 operation system
Orinoco Gold IEEE802.11b PCMCIA wireless NIC

[9] During the test, the parameters of Linux network stack are the default values that are specified when the Linux Operation System is installed. Network Queue Length = 100 packets, Socket buffer = 65536 bytes.

Traffic generator: InterWatch 95000 (IW95000) network analyzer/generator [IW95000]
GN Nettest's InterWatch 95000 (IW95000) provides a combination of modular interfaces and test-support applications for leading technologies including VoIP, MPLS, universal Mobile Telecommunications System (UMTS), QoS measurement, ATM Signaling (UNI, PNNI), and IP Performance and Access technologies. The IW95000 can be connected simultaneously to multiple Ethernet segments, while monitoring network performance (e.g. throughput and latency).

6.1.2 Software

In this test, three software tools are used to record the test metrics. These software tools are developed by the author of this thesis and are listed in appendix A. The software tools include:

Packet NIC delay monitor

This monitor program is designed to record the NIC delay of the IEEE802.11b wireless LAN network card. The NIC delay monitor is patched to the Orinoco wireless LAN driver: "Orinoco.c". In order to export the measured delay value to the Linux user space, a new proc file entry (/proc/mytest) is added to the kernel. Through this proc file entry, the user space application can read the current NIC delay value and save it to the test log file.

Wireless Channel Monitor

One of the main problems in wireless communications is the time varying behavior of the wireless channel. This variability has significant impact on the performance of WLAN. In order to assess and understand this impact, a wireless channel monitor is designed, which records the quality of the wireless channels during the period when packets are transmitted. The wireless tools package [TOU] provides a method to monitor the values of link quality[10], signal level and noise level.

[10] This value indicates the status of the wireless channel condition. It is a vendor specified value.

The link quality is a value reported from the driver of IEEEE802.11b wireless card. Usually users can read this value from the file under /proc/wireless or use command iwconfig to show the status of the specific wireless card. The link quality is a relative value defined by the vendor of the wireless card. Wireless NICs produced by different vendors might have different "readings" of channel quality even in the same channel environment and been used in close proximity. So it will be a different value on different cards even they are working in the same conditions. The wireless cards we used in our testbed are Orinoco Golden cards, manufactured by Lucent. The user guide indicates that the range of the link quality is from 0-90. However in our test, we could only observe the values up to about 40-50 even when there is no the interference in the channel. The link quality drops to 0 if the wireless card works in bad wireless channel conditions

Network Throughput Monitor

This program is based on the library of PCAP package. It captures all the packets arriving or leaving the specified network interface. The packet length and time stamp are recorded for later analysis. We use a filter to identify packets based on their IP header value.

6.2 Evaluation of the Developed IEEE802.11b wireless LAN Testbed in an "Indoor" Environment

This section describes the evaluation test of IEEE802.11b wireless LAN hardware in a typical indoor environment. This test work is to determine how the IEEE 802.11b wireless LAN works in a typical indoor environment. Through this evaluation test, we will understand the needs and limitations of our testbed network and ensure there are no serious performance problems hidden in the low-level hardware that will influence the corrected data. The test cases include peer-to-peer and multi-stations configurations. In order to assess the effect of the time varying wireless channel condition, the test was performed under good wireless channel conditions as well as heavy radio noise interfering conditions.

6.2.1 UDP traffic on Peer-to-Peer configuration in good wireless channel conditions

6.2.1.1 Test objective

This test evaluates the capacity of the IEEE 802.11b PCMCIA wireless LAN in the fixed Peer-to-Peer configuration. The testbed structure is shown in figure 6.1. Figure 6.2 provides the path of the traffic through in network stack of Linux kernel. The test is performed in CBY_B308 (Broadband Wireless and Internetworking Research Lab) of University of Ottawa[11]. The area of this room is about 20 meters * 20 meters. In this room, there are no obstacles such as iron or concrete walls. When the test is performed, we use a wireless LAN site monitor tool (This tool is included in the Orinoco wireless LAN tool kit for Windows 2000) to ensure there are no other active wireless stations in this area except those used in our test. The parameters used in this test are shown in table 6. 1. The metrics that are used to evaluate the performance of the wireless LAN in this study include link quality, throughput, packet loss and NIC delay. The dependence relationships among the metrics and the parameters are shown in table 6.2

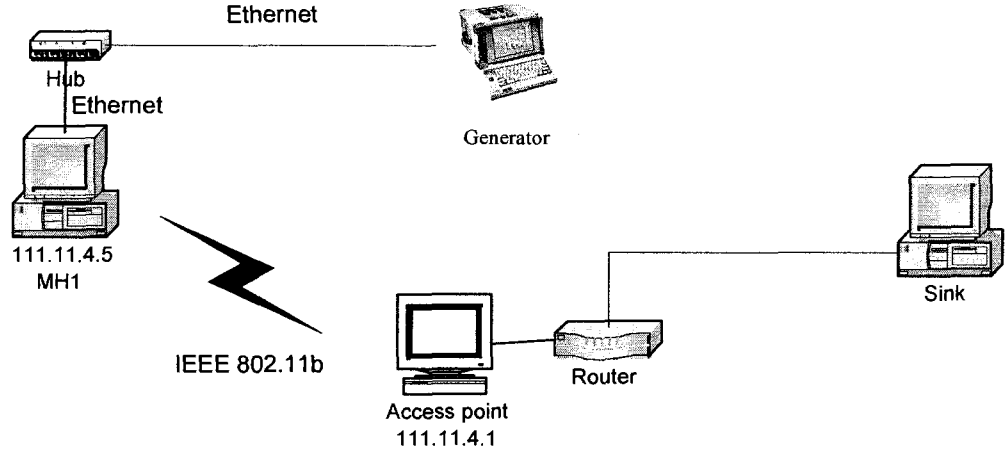


Figure 6.1 Testbed structure of Peer-to-Peer configuration for the experiment where the wireless channel is in good wireless channel condition

[11] All the test cases described in this chapter are performed in the LABb308, University of Ottawa

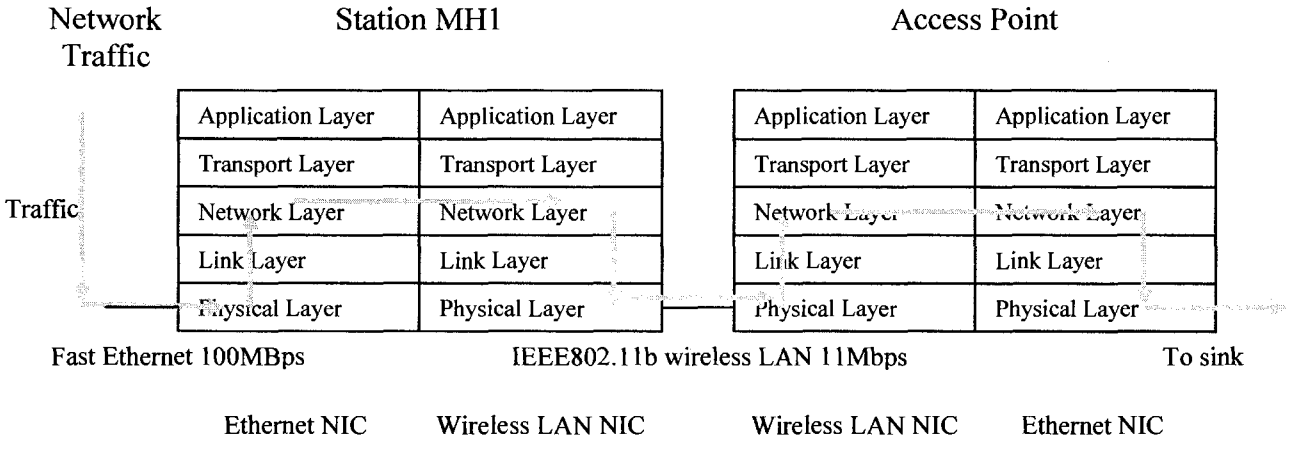


Figure 6.2 Experiment Logic in network stack of Linux kernel

The traffic path in this test is:

- Traffic loading speed/number is set at the Network Traffic Generator
- Number of packets arriving at station MH1 from the generator is measured at the Ethernet NIC on station MH1
- Number of packets sent out from wireless NIC is measured at the wireless LAN NIC on station MH1
- Number of packets arriving at Access Point is measured at the wireless LAN NIC on Access Point
- Packet delay is measured at the MAC layer of the wireless LAN NIC on station MH1
- Throughput is measured on Access Point
- Wireless channel condition is measured at the wireless LAN NIC on station MH1

Table 6.1 Parameters of UDP test on Peer-to-Peer configuration in good wireless channel condition

Parameters name	Values	Description
Traffic generating speed	CBR 1,5,11 Mbps	The different generating speed values are used to evaluate the performance of IEEE802.11b wireless LAN under different traffic loads.
IP packet length	100, 500, 1000,1500 bytes	The different packet lengths are used to evaluate the influence of packet length on the performance of IEEE802.11b wireless LAN
Network NIC transmission speed	11 Mbps	The maximum capacity of IEEE802.11b wireless LAN is evaluated

Table 6.2 Measured metrics of UDP test on Peer-to-Peer configuration in good wireless channel condition

Measurement metrics	Depending on:	Description
Link quality[12]	This value is vendor specified. For Orinoco NIC, it represents SNR (Signal Noise Ratio)	This value represents the wireless channel condition. The wireless LAN NIC reports this value directly to wireless NICs driver.
Throughput	Traffic load, packet length, transmission	This is throughput capacity of IEEE802.11b wireless LAN

[12] The Link quality read from wireless card driver is a vender depended value. For the Orinoco Gold card, which is used in this test, the value is changed from 0 to 90.

	speed, link quality	
Packet loss	Traffic load, packet length	This metric demonstrates the effect of varying amounts of packet error probability on packet loss
NIC delay	Traffic load, link quality, packet length	This is the measured NIC delay under of IEEE802.11b wireless LAN

6.2.1.2 Test procedure

In this test, CBR UDP traffic is sent from fixed wireless station MH1 (111.11.4.5) to sink through Access Point (111.11.4.1). The transmission speeds of the wireless NICs are set as 11Mbps. The InterWatch 95000 generator generates the traffic at speed of 11Mbps. The sizes of UDP packets are set as 100, 500, 1000 and 1500 bytes. The test procedure is repeated 5 times and 100,000 packets are transmitted at each test.

6.2.1.3 Test results and analysis

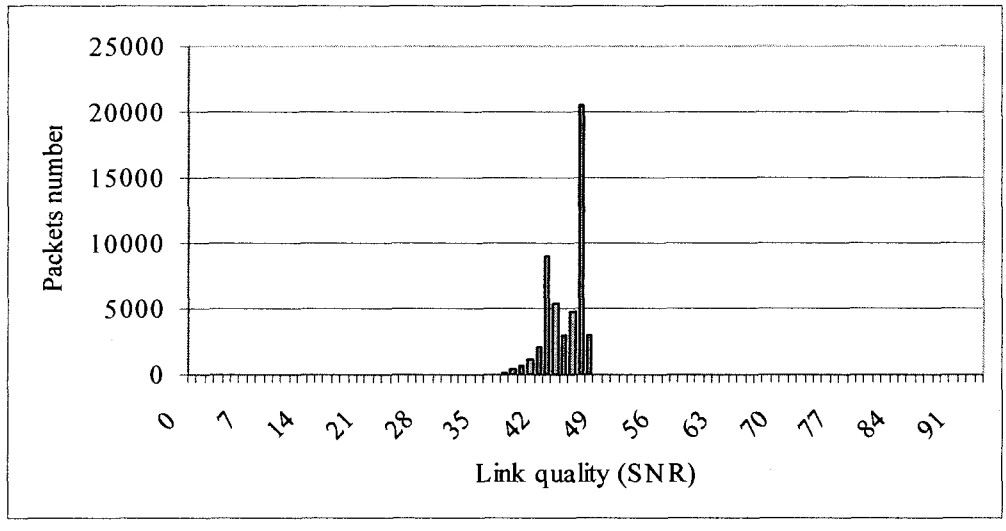


Figure 6.3 The distribution of link qualities with packet numbers in the process of UDP test on Peer-to-Peer configuration. Test environment: Indoor environment without radio noises

Figure 6.3 provides the link quality values in the LAB308b, University of Ottawa. There are no radio noise and obstacles in this Lab. The figure shows that all the link quality values are

located in the range (38, 47). The results mean that the value of link quality is stable around 40 in the good wireless channel condition.

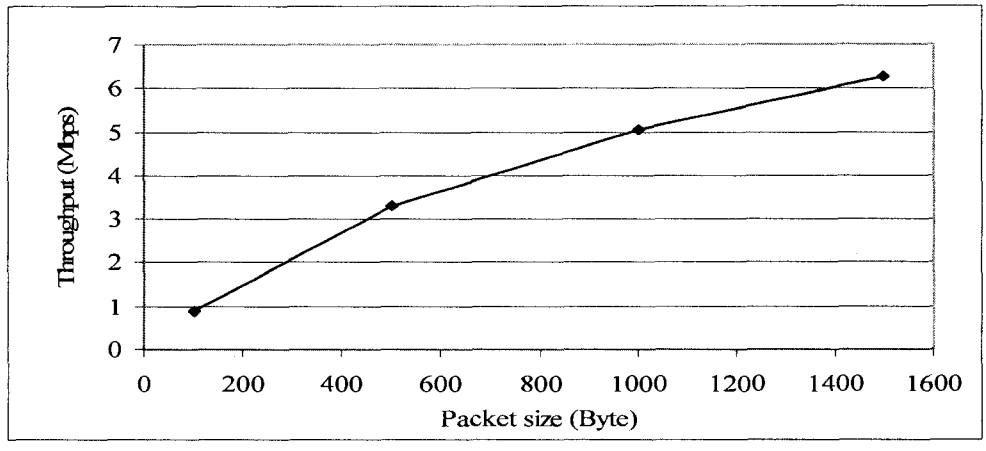


Figure 6.4 Average throughputs for CBR UDP traffic on Peer-to-Peer configuration, test environment: Indoor environment without radio noises and obstacles

In figure 6.4, the throughputs of Peer-to-Peer UDP traffics with different packet sizes are displayed. From this figure, we conclude that the throughput increases with the packet size. When the data rate of the Orinoco wireless LAN NIC is configured as 11Mbps and the packet size is set as 1500 bytes, the highest throughput for Peer-to-Peer UDP traffic transmitted is about 6.3Mbps in the indoor environment with good wireless channel condition.

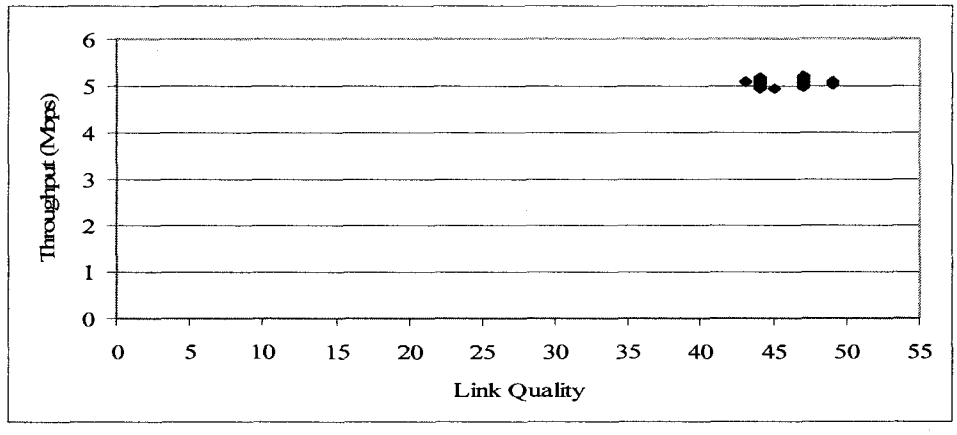


Figure 6.5 Correlation between Link quality and throughput for CBR UDP traffic on Peer-to-Peer configuration. Test environment: Indoor environment without radio noises and obstacles, transmission distance = 5M

The figure 6.5 provides the correlation between the link quality and throughput. We make the link quality values as the X-axis and throughput values as Y-axis. So every point in this figure represents a *(Link quality, Throughput)* pair. From this figure, the correlation between Link quality and throughput distribution is clearly displayed. The figure shows that the link quality values change between 40 and 50 when there are no radio noises and obstacles existed in the indoor test environment. With this good wireless channel conditions, the throughput values of Orinoco wireless LAN NIC are about 6.1Mbps when packet size is 1000 bytes and traffic generating speed is 11Mbps and do not change dramatically.

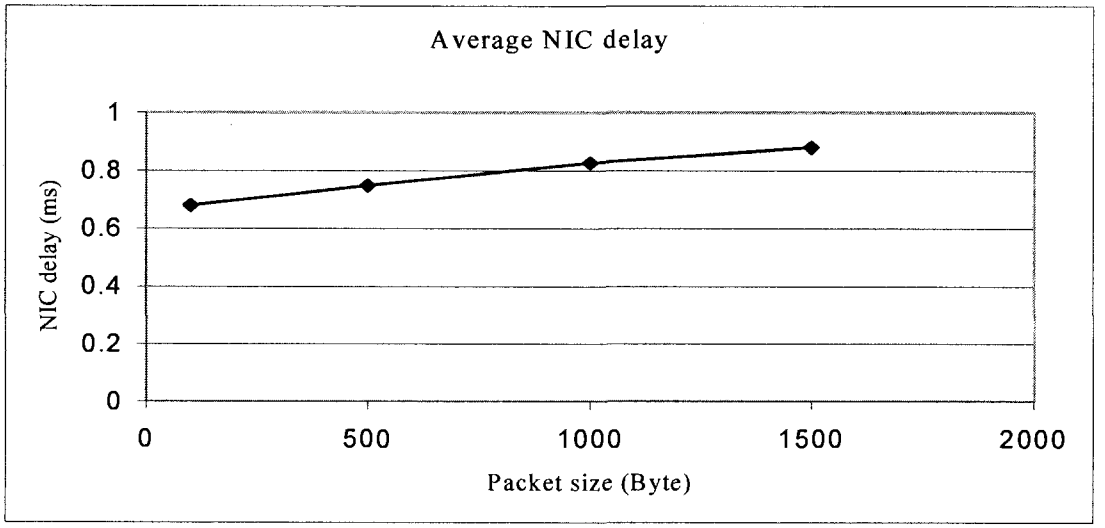


Figure 6.6 NIC delay for CBR UDP traffic in Peer-to-Peer configuration, test environment: Indoor environment without radio noises and obstacles

The figure 6.6 displays the average NIC delay of packets with different sizes measured by the packet delay monitor (The monitor tool is introduced in section 6.1.2). In this Peer-to-Peer test, only one traffic is transmitted across the wireless LAN and there are no radio noise or signal obstacles in the test environment, so the average NIC delays are very small because no collision and retransmission.

6.2.2 UDP traffic on multi-station configuration in good wireless channel conditions

6.2.2.1 Test objective

This test evaluates the capacity of IEEE 802.11b PCMCIA wireless LAN in the fixed multi-stations configuration. The testbed structure is showed in figure 6.7. As we know, IEEE 802.11 wireless LAN uses the contention based CSMA/CA protocol as MAC layer. This test is to find out how this protocol works in a real multiple stations (users) environment. The parameters used in this test are showed in table 6.3. The metrics that are used in this test case are showed in table 6.4.

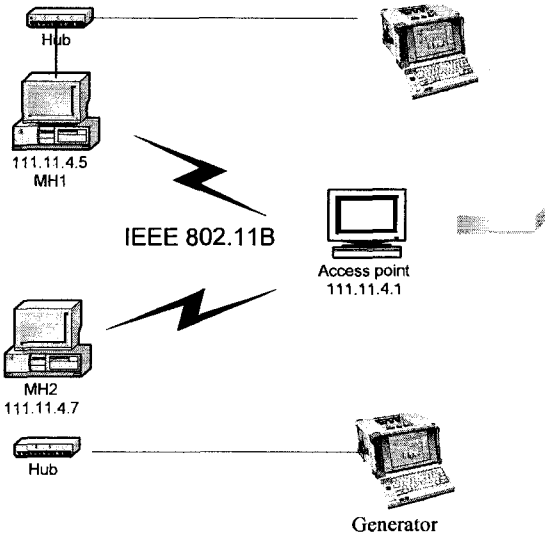


Figure 6.7 Testbed structure of multiple stations configuration in good wireless channel conditions

Table 6.3 Parameters of UDP test on Peer-to-Peer configuration in good wireless channel condition

Parameters name	Values	Description
Traffic generating speed of traffic1	CBR 3.5 Mbps	The actual throughput of this traffic is measured to evaluate the effect of background traffic with different generating speeds

Traffic generating speed of traffic2 (Background traffic)	CBR 2, 4,6,8 Mbps	The different generating speeds of background traffic are used to evaluate the performance of IEEE802.11b wireless LAN under different traffic loads.
IP packet length	1000 bytes	The packet length is a constant value because we focus on the effect of contention among multiple stations
Wireless LAN NIC transmission speed	11 Mbps	The maximum capacity of IEEE802.11b wireless LAN is evaluated

Table 6.4 Measured metrics of UDP test on Peer-to-Peer configuration in good wireless channel condition

Measurement metrics	Depending on:	Description
Total throughput of wireless LAN	Traffic load of background traffic	This metric demonstrates the actual total throughput capacity of IEEE802.11b wireless LAN under different traffic load
Throughput of traffic transmitted on station MH2 (Target traffic)	Traffic load of background traffic	This is the actual throughput for one station when the IEEE802.11b wireless LAN has different traffic load
NIC delay	Traffic load, link quality	This is the NIC delay of IEEE802.11b wireless LAN with different traffic loads

6.2.2.2 Test procedure

The Testbed includes two stations and one Access Point (AP). CBR UDP traffic (Traffic 1) is sent from station MH2 (111.11.4.7) to sink1 through AP (111.11.4.1). At the same time, a

background CBR UDP traffic (Traffic 2) is sent from station MH1 (111.11.4.5) to sink2. The transmission speeds of the wireless LAN NICs are set as 11Mbps. The InterWatch 95000 generators generate both of the traffics. The generating speed of traffic 1 is 3.5 Mbps and the generating speed of background traffic is 2, 4, 6, 8 Mbps in different tests. The size of IP packet is set as 1000 bytes.

6.2.2.3 Test results and analysis

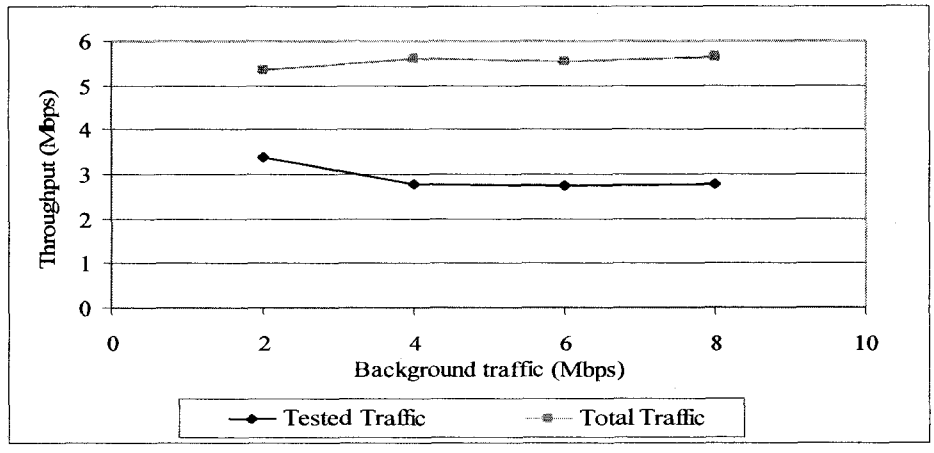


Figure 6.8 Average throughputs for UDP traffic on multiple stations configuration with different background traffic loads, test environment: Indoor environment without radio noises and obstacles

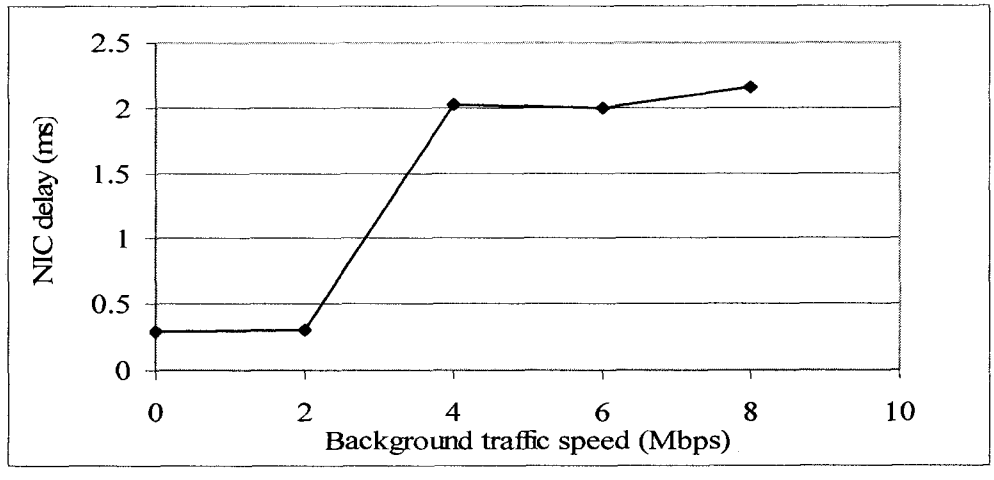


Figure 6.9 Average NIC delay for UDP traffic on multi-station configuration with different background traffic loads, test environment: Indoor environment without radio noises and obstacles

From the figure 6.8 and 6.9, we can observe that the number of simultaneous users has obvious effect on the throughput and NIC delay. Although the IEEE802.11b standard specifies that the transmission data rate of wireless LAN can be set as 11 Mbps, the maximum total throughput capacity of the multiple stations wireless LAN subnet is about 6.5 Mbps to 6 Mbps when the packet size is 1000 bytes.

The other observation is that the maximum throughput depends on the network load and the number of simultaneous users. When the background traffic is 2 Mbps, the total traffic load is 6.5 Mbps (3.5 + 2), which is less than maximum transmission capacity of wireless LAN. So the tested traffic can get the ideal speed (about 3.5 Mbps) and the NIC delay is lower (figure 6.9). However, with the background traffic is increased to 4 Mbps, the total traffic load is 7.5 Mbps, which is less than maximum capacity of wireless LAN. As the result, the tested traffic only can get the speed of about 2.8 Mbps and the NIC delay is bigger. However, even with the background, traffic increases to 6 Mbps and 8 Mbps, the tested traffic still can get the speed of about 2.8 Mbps and the NIC delay increased little. The phenomena showed that under CSMA/CA protocol, the maximum throughput one station can get is dependent on the number of active traffics. If all the traffics are generated in a rate bigger than the average speed (which is equal the maximum capacity divided by the number of traffic), all the traffics would be transmitted at the average speed, whatever how fast the traffic is generated.

6.2.3 The Influence of radio noise on Peer-to-Peer UDP traffic

6.2.3.1 Test objective

We know that the wireless channel condition is tending to be influenced by the obstacle, distance and radio noise. In out experiment, we investigated a special case, in which two IEEE 802.11b wireless cards are installed side by side to one PC. This test case shows how the interference between two wireless cards will influence the performance. Figure 6.10 shows the testbed configuration. The PC, on which two cards are installed, works as a gateway, which communicates with Access point in WLAN 1(111.11.4.0) and also connect

with roamer through another WLAN (192.168.1.0). We have configured the two WLANs to work in two separate frequencies, but we still observed the interference in the test results.

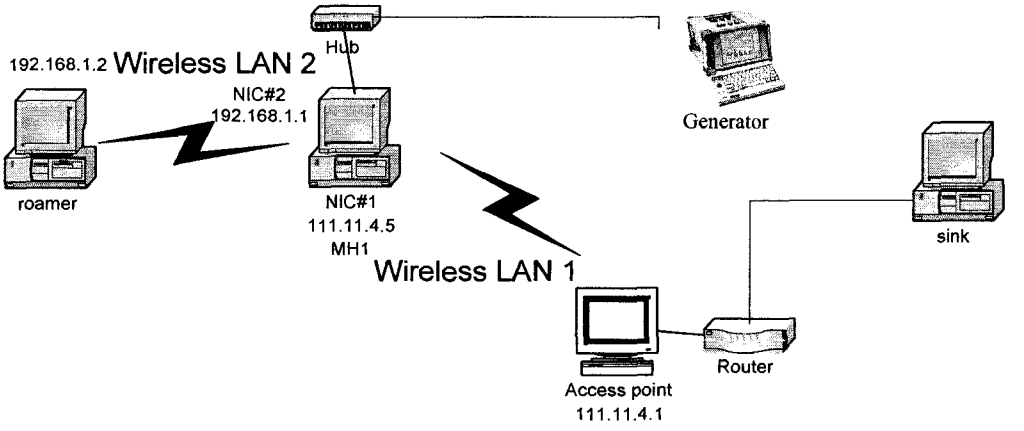


Figure 6.10 Testbed structure of Peer-to-Peer configuration in wireless environment where the radio noise exists

The parameters used in this test are showed in table 6.5. The metrics that are used to evaluate the performance of wireless LAN in this test case are showed in table 6.6.

Table 6.5 Parameters of UDP test on Peer-to-Peer configuration in bad wireless channel condition

Parameters name	Value	Description
Traffic generating speed of traffic transmitted by NIC #1 on station MH1	CBR 2 Mbps	The actual throughput of this traffic is measured to evaluate the effect of radio interference from the other active wireless NIC installed on the same station (MH1)
Traffic generating speed of traffic transmitted by NIC #2 on station MH1	CBR 3 Mbps	The actual throughput capacity of wireless LAN NIC #1 is evaluated when the transmission signal of NIC #2 make the interference
IP packet length	1000 Bytes	The packet length is a constant value because we focus on the effect of signal interference between two wireless LAN NICs

Network NIC transmission speed	11 Mbps	The maximum capacity of IEEE802.11b wireless LAN is evaluated

Table 6.6 Measured metrics of UDP test on Peer-to-Peer configuration in bad wireless channel condition

Measurement metrics	Dependency	Description
Link quality	This value is vendor specified. For Orinoco NIC, it represents SNR (Signal Noise Ratio)	This value represents the wireless channel condition. It depends on the signal level and noise level. The wireless LAN NIC reports this value directly to the driver
Throughput of wireless NIC #1 of station MH1	Link quality	This is the actual throughput capacity of a IEEE802.11b wireless LAN station when it works in bad wireless condition
NIC delay of wireless NIC #1 of station MH1	Link quality	This is the actual transmission of a IEEE802.11b wireless LAN station when it works in bad wireless condition

6.2.3.2 Test procedure

In this test, two wireless LANs are set up: wireless LAN 1 works on channel 3 (2.422 GHz) and wireless LAN 2 works on channel 8 (2.447 GHz). The station MH1 works as a joint point on which two wireless LAN NICs are installed. The NIC #1 and NIC #2 belong to different wireless subnets (111.11.4.0 at 2.422 GHz and 192.168.1.0 at 2.447 GHz). A 2 Mbps CBR UDP traffic is transmitted from MH1 to AP (111.11.4.1) and a 3 Mbps CBR UDP traffic is transmitted from MH1 to roamer at the same time.

6.2.3.3 Test results and analysis

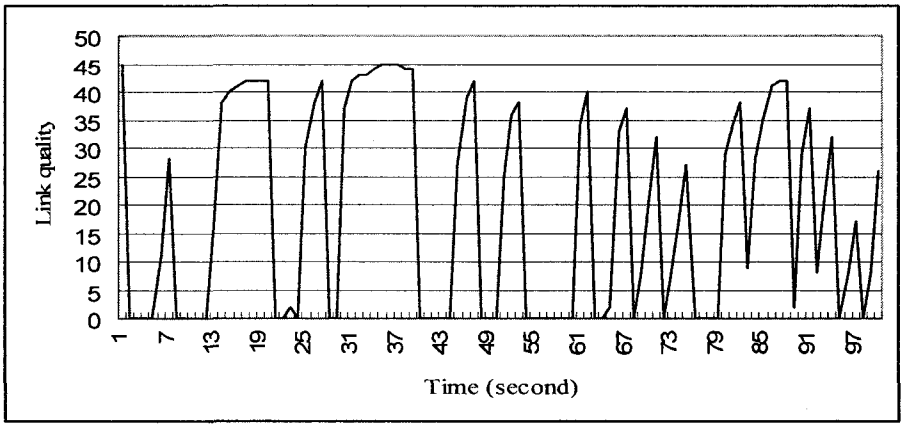


Figure 6.11 The changing curve of bandwidth versus time for Peer-to-Peer UDP traffic

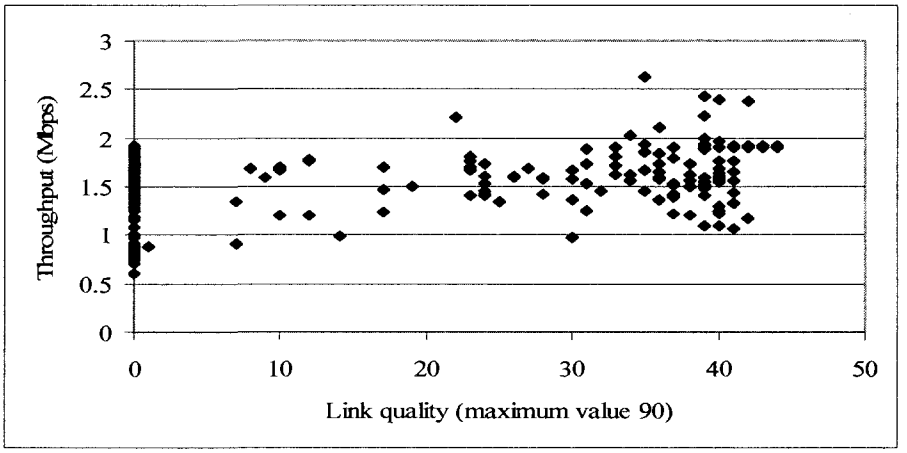


Figure 6.12 Aggregate link quality and throughput distribution for Peer-to-Peer UDP traffic

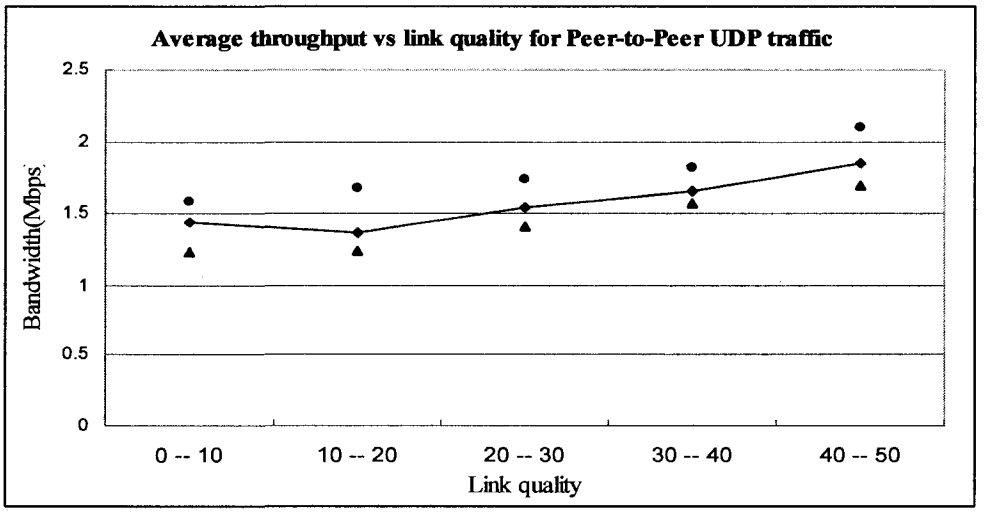


Figure 6.13Average throughput versus link quality for Peer-to-Peer UDP traffic

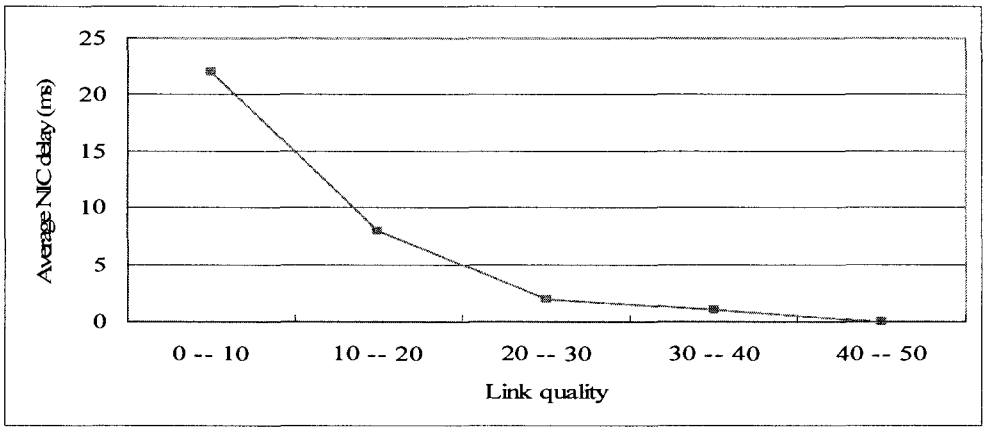


Figure 6.14 Average NIC delay versus link quality for Peer-to-Peer UDP traffic

Figure 6.11 shows the changing curve of the bandwidth in the test. Figure 6.12 shows the changing curve of the link quality in the same test run. We can observe that both the link quality and bandwidth varies dramatically during the test under the influence of the interference between two active side-by-side installed WLAN cards. Figure 6.11 also shows the effective bandwidth has been cut to about one third of the maximum capacity of WLAN.

Figure 6.13 provides the correlation between link quality and throughput. In Figures 6.13, every node corresponds the measured bandwidth and link quality at one time. The wide spreading distribution of the nodes indicates the interference between WLAN cards has a big influence on the performance of the WLAN, which becomes unstable and user cannot get a clear linear mapping curve between link quality and bandwidth in this case. Figure 6.14 shows the curve of average bandwidth versus link quality. We marked the varying range of the bandwidth corresponding to the link quality in the five repeated test runs. The results confirm that the average bandwidth only has a slight difference corresponding to the different link quality range.

6.2.4 The Influence of radio noise on multiple stations UDP traffic test

6.2.5 Test objective

The above test shows that an IEEE802.11b wireless NIC will be influenced by other wireless NIC nearby. Unfortunately, this interference will extend to whole wireless LAN and bring troubles to other stations even when these stations work in good wireless channel conditions. This test case is to evaluate the effect of radio noise interference on multiple stations wireless LAN subnet. The testbed structure is showed in figure 6.15.

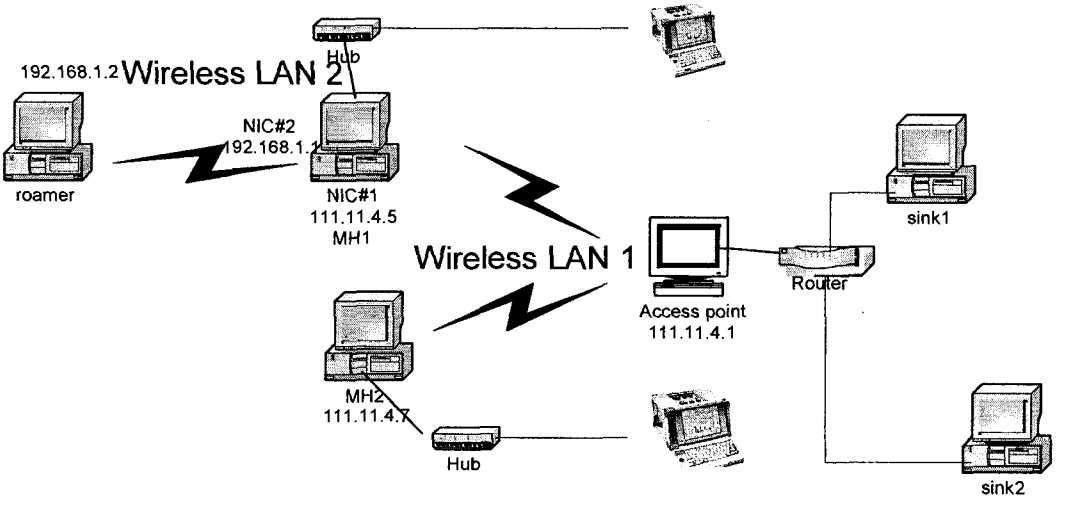


Figure 6.15 Testbed structure of multi-stations configuration in wireless environment where the radio noise exists

The parameters used in this test are showed in table 6.7. The table 6.8 shows the metrics that is used in this test case.

Table 6.7 Parameters of UDP test on Peer-to-Peer configuration in bad wireless channel condition

Parameter name	Value	Description
Traffic generating speed of traffic transmitted by NIC #1 on station MH1	CBR 2 Mbps	The actual throughput of this traffic is measured to evaluate the effect of radio interference from the other active wireless NIC installed on the same station (MH1)

Traffic generating speed of traffic transmitted by NIC #2 on station Mh1	CBR 1.8 Mbps	The actual throughput of wireless LAN NIC #1 is evaluated when the transmission signal of NIC #2 make the interference
Traffic generating speed of traffic on station MH2	CBR 3 Mbps	The actual throughput of this traffic is measured to evaluate the effect of a station working in bad wireless channel condition on other stations
IP packet length	1000 bytes	The packet length is a constant value because we focus on the effect of signal interference between two wireless LAN NICs
Network NIC transmission speed	11 Mbps	The maximum capacity of IEEE802.11b wireless LAN is evaluated

Table 6.8 Measured metrics of UDP test on Peer-to-Peer configuration in bad wireless channel condition

Measurement metric	Dependency	Description
Throughput of station MH2 (Traffic 1)	Link quality	This is the actual throughput capacity of station when one of the stations belonging to the same wireless LAN subnet works in bad wireless channel condition
Throughput of wireless NIC #1 of station MH1 (Traffic 2)	Link quality	This is the actual throughput capacity of a IEEE802.11b wireless LAN station when it works in bad wireless condition
NIC delay of station MH2 (Traffic 1)	Link quality	This is the actual NIC delay of station when one of the stations belonging to the same wireless LAN subnet works in bad wireless channel condition
NIC delay of wireless NIC #1 station MH1 (Traffic 2)	Link quality	This is the actual transmission of a IEEE802.11b wireless LAN station when it works in bad wireless condition

6.2.5.1 Test procedure

During this test, a 2 Mbps background CBR UDP traffic (traffic 2) is transmitted from NIC#1 of station MH1 to sink2 through AP (111.11.4.1) and a 1.8 Mbps CBR UDP traffic is transmitted from NIC#2 of station MH1 to roamer (192.168.1.2). At the same time, a 3 Mbps CBR UDP traffic (traffic 1) is transmitted from MH2 to sink1 through AP. The wireless LAN NIC of station MH2 works in good wireless channel condition because there are no radio noises near station MH2.

6.2.5.2 Test results and analysis

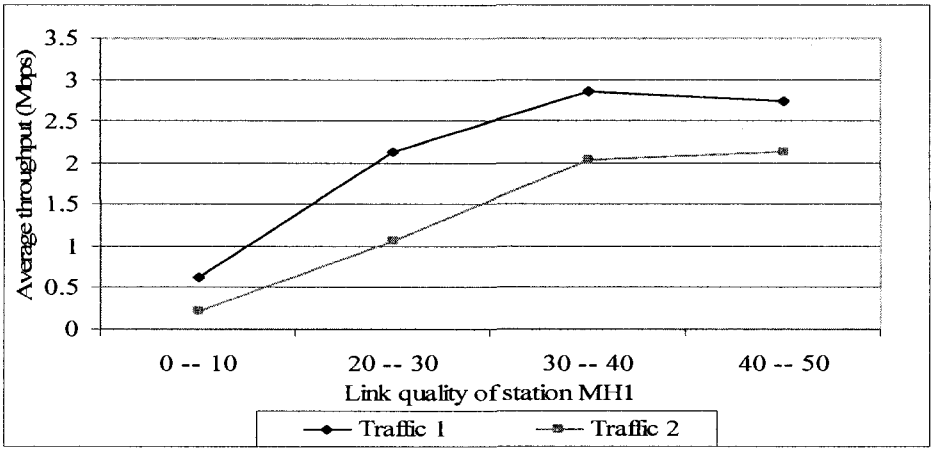


Figure 6.16 Average throughput versus link quality for multi-station UDP traffic

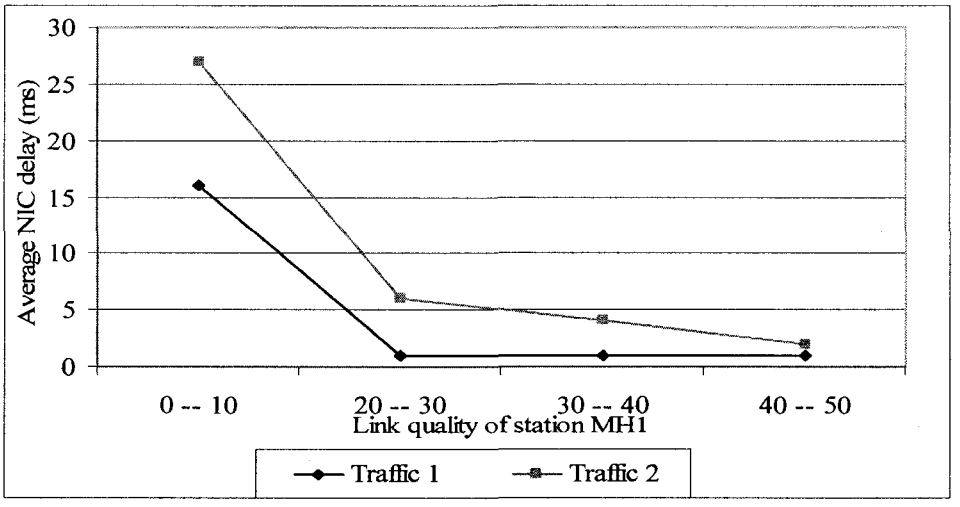


Figure 6.17 Average NIC delay versus link quality for multi-station UDP traffic

The figure 6.16 and figure 6.17 show that the interference of radio noise for one station extends to the other stations in the same wireless LAN. When the link quality of station

MH1 becomes bad because of the interference from two simultaneous active wireless NICs, the observed throughputs of both traffic (from station MH1 and MH2 to AP) decrease, too. This means that the station MH1 consumes a lot of resource because of the large amount of retransmission due to bad wireless channel condition. As the result, the NIC delay of traffic2 also increased obviously. This means the retransmissions of traffic from station MH1 congest the whole wireless LAN. The wireless NIC on station MH2 has to wait for longer time to catch the transmission opportunity.

Chapter 7

Evaluation of Distributed Network Congestion Control Implementation Prototype

This chapter presents the performance evaluation of the proposed distributed network congestion control scheme. As mentioned earlier, we have implemented it using standard PC, running on the RedHat Linux 7.3 version operating system. The tests following aim to assess the effectiveness of the proposed scheme to support QoS, as well as help us to understand the function of system parameters.

7.1 Simultaneous Transmission of Expedited Forwarding and Best Effort Traffic Stream

In this set of experiments, the EF and AF traffic streams are transmitted. Both of them are CBR UDP streams. The testbed topology is shown in Figure 7.1.

In this set of experiments, the EF/BE traffic streams are transmitted through the IEEE802.11b wireless LAN. The distributed network congestion control schemes are deployed in the Linux kernel of the mobile hosts (MH1 and MH2) and Access point for the purpose of providing QoS support for the EF traffic flows. When the network becomes congested, the QoS scheme dynamically reduces the data rate of BE traffic flows, preserving bandwidth resource for the EF traffic flows. Therefore, the delay and delay jitter of the EF traffic flows is reduced. The packet loss rate of EF traffic is expected to be reduced, while the average bandwidth consumed by the EF traffic flows is expected to increase. Of course, this comes at the expense of leaving less resource to the BE traffic flows.

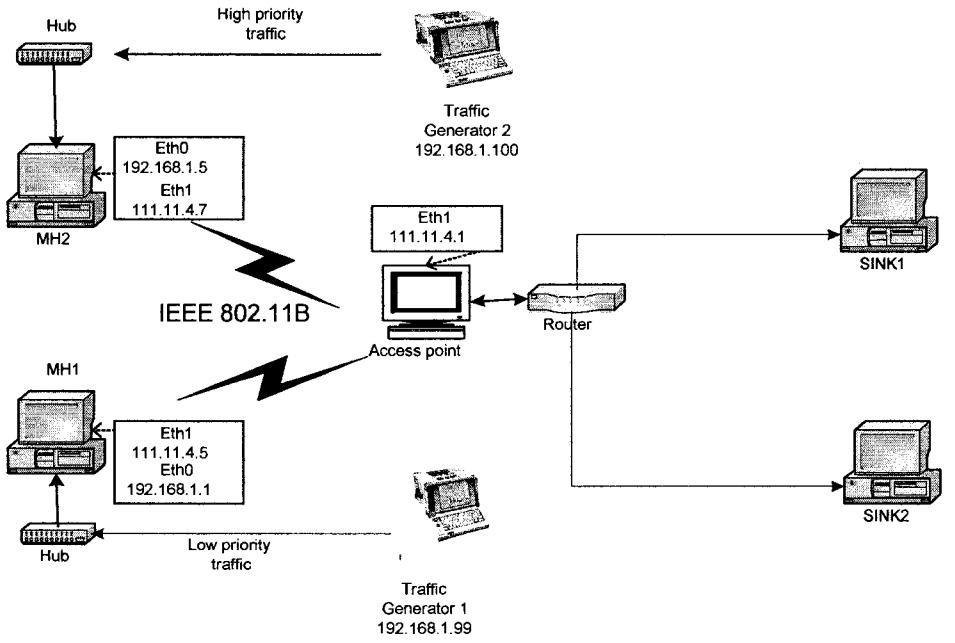


Figure 7.1 Architecture of test network

As we mentioned, in this set of experiments, the system's performance is evaluated using CBR UDP traffic sources. The traffic generator 2 generates the EF flow, while the station MH2 transmits it through the wireless LAN. Traffic generator 1 and MH1 perform the same function for the BE flow. For every test, a constant number of UDP EF packets (200,000 packets with size = 1000 bytes payload) are generated by the traffic generator 2 and are transmitted across through the wireless LAN.

The measured QoS metrics are packet loss rate, average NIC delay, NIC delay jitter and average reserved bandwidth of the EF traffic flows. For comparison purposes, we also show the average bandwidth consumed by BE traffic flows. The metrics are calculated as follows:

1. Packet loss rate

In the test, assume that M UDP EF packets are transmitted. At the access point, N UDP EF packets are captured by ethereal. The packet loss rate is calculated as (M- N)/M.

2. Average NIC delay

The definition of NIC delay was introduced in section 4.2. In section 5.2.2, we described the implementation of the NIC delay monitor, using Linux /proc system. When using a PC running at 1.8 Ghz CPU, the accuracy of the NIC delay is the main frequency of the CPU used in the PC is 1.8 Ghz, which means that the time precision we can get is $1/1.8\times10^{9}\ \mathrm{second} = 0.56\times10^{-9}\ \mathrm{second} = 0.56\ \mathrm{nanoseconds}$.

3. NIC delay jitter

The NIC delay jitter is calculated as the Standard Deviation of the NIC delay. The standard deviation is a measurement of how widely values are dispersed from the average value (the mean value of the sample). The standard deviation is calculated using the STDEV function from MS EXCEL. The function is :

$$\sqrt{\frac{n\sum x^2-(\sum x)^2}{n(n-1)}} \tag{7.1}$$

where n is the number of samples and x_i represents the value of sample "i".

4. Bandwidth

The bandwidth is calculated using a 1 second time window. In the access point, the arrived UDP EF and BE packets are separately captured by the Bandwidth monitor using the pcap library [PCAP] and the two numbers of arrived packets are delivered. The EF (BE) bandwidth is the number of EF (BE) packets arrived within this 1 Second period.

In this section, we investigate the effect of the following system parameters:

- Increasing factor C_{in}^{i}
- Decreasing factor r_{de}^{i}
- Rate-updating interval T

The above system parameters have been defined and discussed in detail in section 4. 2. With the QoS schedule applied, station MH2, through which the EF traffic is transmitted, broadcasts a data rate decreasing request packet when the NIC delay measured at the driver of wireless LAN card exceeds the delay threshold d_{th}. After station MH1 receives the packet, it decreases the data rate of its BE traffic multiplicatively by the decreasing factor r_{de}^{i}, saving bandwidth for the high priority EF traffic. On the other hand, when the measured NIC delay is lower than the desired delay d_{th}, station MH2 will broadcast a data rate increase request packet to inform station MH1 to increase its data rate. The NIC delay is measured at the rate of $1/T$.

In order to detect congestion at the early stage, the desired NIC delay threshold d_{th} and rate-updating interval T should be small. If the rate-updating interval T of data rate control is not small enough, the system will not be able to capture the changes fast enough to prevent serious deterioration due to network congestion. In this case, while the network already been congested, massive packets losses will occur and the delay will increase considerably, seriously impairing the applications serviced by the EF traffic. On the contrary, if the traffic volume decreases, bandwidth resource will be wasted if the increase in BE traffic bandwidth does not occur fast enough. It is evident that the value of the performance related parameters should be chosen properly in order to avoid resource losses and/or performance deterioration of the serviced applications.

7.2 Test Result and Analysis

7.2.1 Test Case1: Performance with 3Mbps CBR UDP traffic at EF

Table 7.1 presents the system parameters of test case 1. The objective of this test case is to evaluate the performance of the proposed QoS scheme. The data rate of the wireless card is set as 5.5Mbps. The data rate of the EF traffic is fixed at 3 Mbps, while the BE traffic is also CBR UDP. The experiments are conducted with the BE traffic set at 0, 0.5, 1, 2, and 3 Mbps. The desired NIC delay is set as 1.2ms. According the test result in section 4.2.2, in the case of the data rate of wireless card is set as 5.5Mbps, if the measured NIC delay is bigger then

1.2ms, the network is over load and the traffic congestion happened. The test case is repeated 5 times. During each test, 40,000 EF packets are sent every time.

Table 7.1 Configuration of parameters and traffic sources used in test case 1.

Parameter/ traffic source names	**Value**
Decreasing factor (r_{de}^{i})	0.25
Increasing factor (C_{in}^{i})	50 Kbps
Desired delay (d_{th})	1.2 ms
Data rate-updating interval (T)	200 ms
EF traffic	3 Mbps CBR UDP
BE traffic	0, 1, 2 or 3 Mbps CBR UDP

Figures 7.2 to 7.7 provide the average NIC delay, delay jitter, packet loss rate and average bandwidth of EF traffic/BE traffic versus BE background traffic volume, respectively. Figures 7.8 to 7.10 present the CDF and PDF packet loss rate distribution of EF/BE traffic. The average bandwidth of BE traffic is also shown in Figure 7.13.

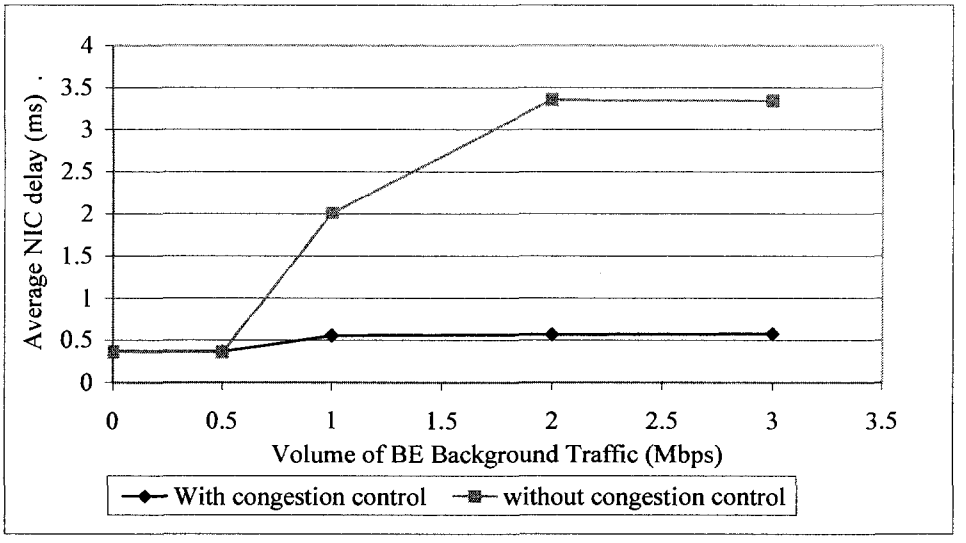


Figure 7.2 Average NIC delay as experienced by EF traffic versus BE background traffic volume

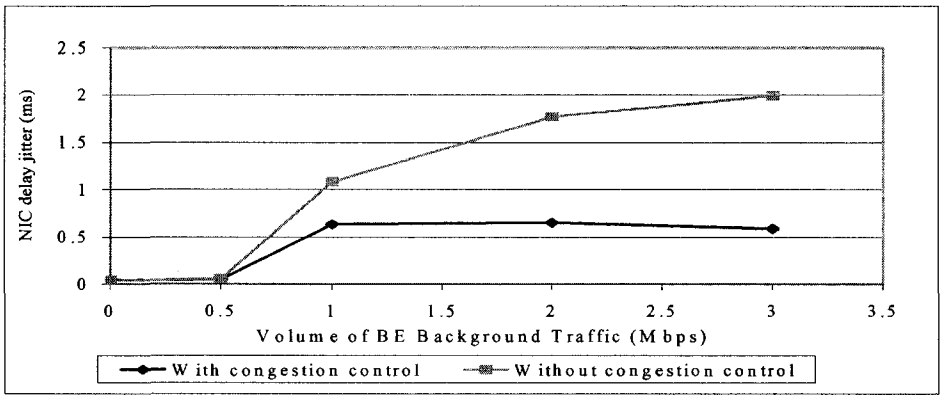


Figure 7.3 Average NIC delay jitter as experienced by EF traffic versus BE background traffic volume

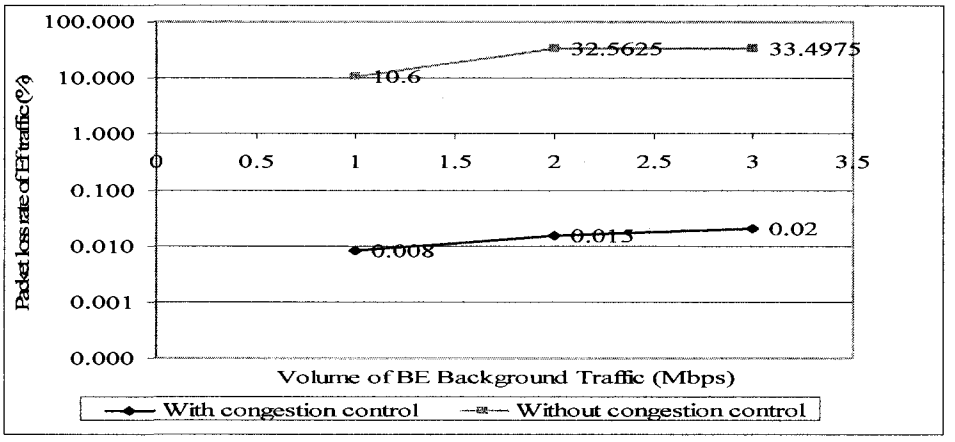


Figure 7.4 Packet loss rate of EF traffic (logarithm scale) versus BE background traffic volume

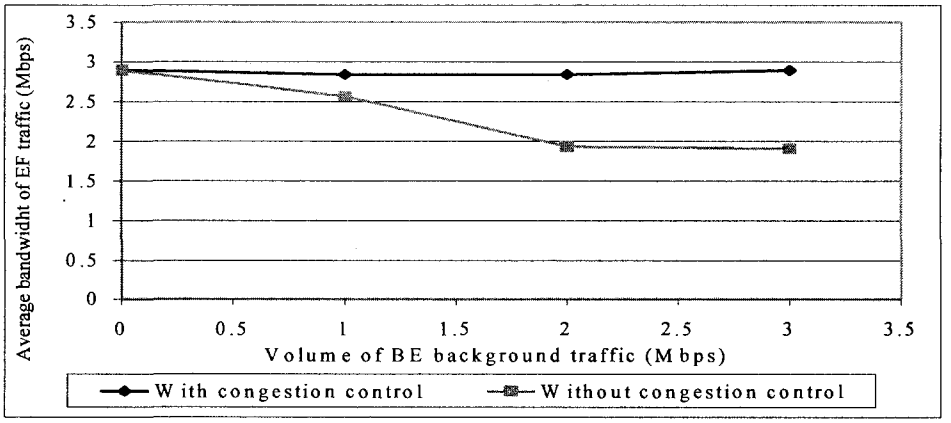


Figure 7.5 Average bandwidth consumed by the EF traffic versus BE background traffic volume

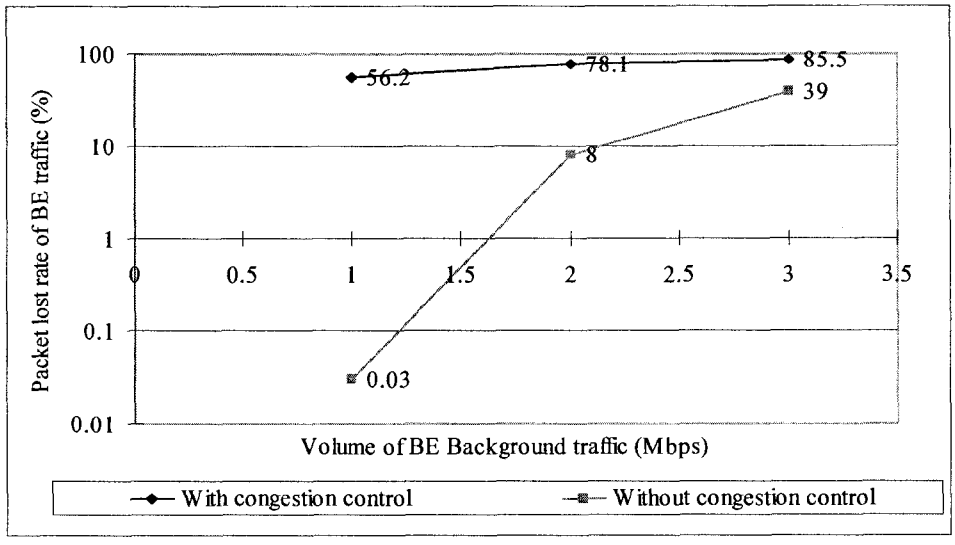


Figure 7.6 Packet loss rate of BE traffic (logarithm scale) versus BE background traffic volume

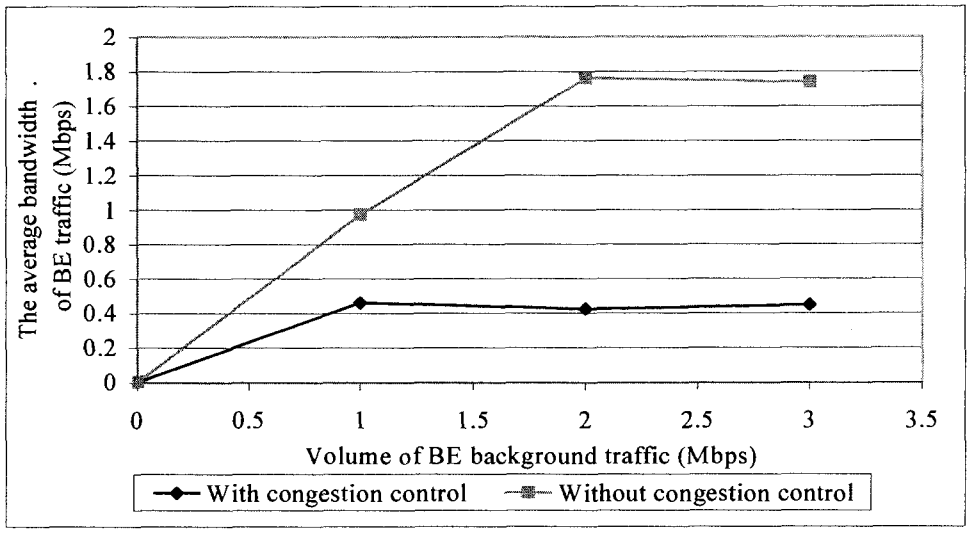


Figure 7.7 Average bandwidth consumed by the BE traffic versus BE background traffic volume

Without the active QoS scheme in place, the EF traffic stream has to compete for bandwidth with the BE traffic based on the best effort DCF mechanism. As the generating speed of BE traffic is increased, the EF traffic gets less share of the bandwidth resource. In contrast, when the QoS schedule is employed, priority is given to the EF traffic. The remaining bandwidth resource is used for the BE traffic.

Figures 7.2 to 7.7 indicate that without the QoS schedule, the increasing data rate of the BE traffic causes longer average NIC delay, larger NIC delay jitter, and higher packet loss to EF traffic and takes bandwidth away from the EF traffic stream. An interesting observation coming from figure 7.2 and figure 7.3 is that the average delay and delay jitter are very close in the value. The reason for this result is that the NIC delay is measured in a very high time accuracy. The measured NIC delay values are very small values (usually 0.5ms) without network congestion. When network congestion happens, the NIC jumped to a big value up to 10 ms. Although the congestion control scheme is able to resolve the network congestion by reducing the bandwidth of BE traffic, the existence of the big NIC delay value makes the jitter value big. Observing Figure 7.4, it becomes evident that the packet loss rate of the system without our QoS scheme is in the range or higher than 10^{-1}, even for 1 Mbps BE traffic volume. This indicates that the network, without QoS, is incapable to even support voice, which is considered one of the tolerant applications to packet losses.

In contrast, when the proposed QoS scheme is applied, as the curve displayed in Figure 7.2, the NIC delay of the EF traffic remains at a small value (around 0.5ms), regardless of what the data rate of the BE traffic is. The results of Figure 7.3 indicate that the delay jitter of the EF traffic remains small as well (approximately 0.6ms) regardless of the data rate of the BE traffic. The packet loss rate of the EF traffic is shown in Figure 7.4. It is evident that the packet loss rate remains 10^{-3} (it is actually close to10^{-4}), even when the BE traffic load becomes high. This confirms that despite of the fact that our scheme is simple and easy to be implemented, it is able to support real time applications, such as voice and video. By comparing figures 7.5 and Figure 7.7, we can see that the average bandwidth of the EF traffic is approximately a constant, around 3 Mbps, which is almost the same as its transmission

rate, and independent of the transmission rate of the BE traffic. The average bandwidth of BE traffic is much lower than its transmission rate and independent of its transmission rate. With the QoS scheme active, its actual bandwidth is determined by the available network resource after satisfying the EF traffic. Therefore, from Figure 7.2 to Figure 7.7, we can see that our QoS schedule works very well.

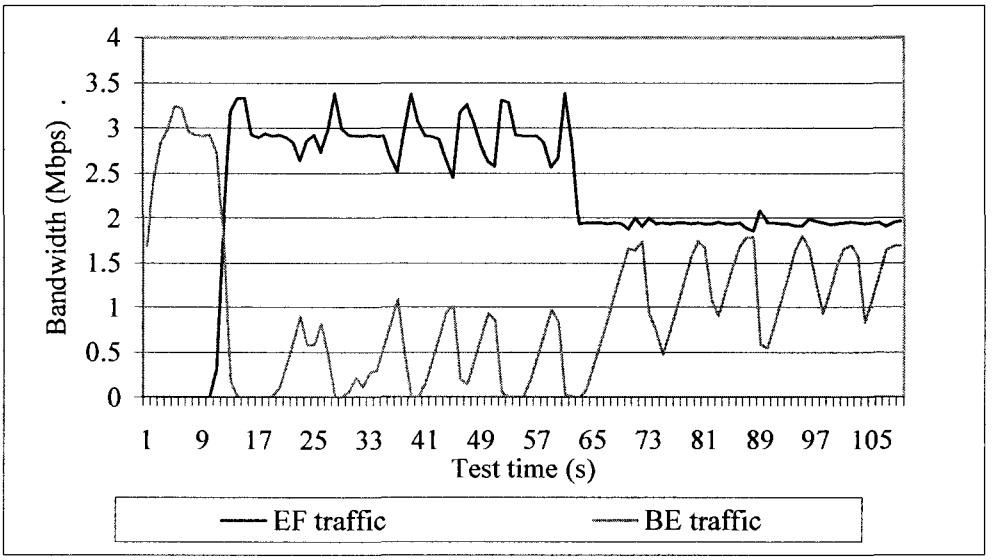


Figure 7.8 Transient behavior of the bandwidth consumption of EF/BE traffic bandwidth

The transient behavior figure – Figure 7.8 -- provides an insight on how the network congestion control mechanism works. From time 0, traffic generator 1 starts a UDP CBR traffic at speed of 3 Mbps, which is forwarded by station MH1 and transmitted to Sink1 as the BE traffic stream. At the 8th second, traffic generator 2 starts another 3Mbps UDP CBR traffic, which is forwarded at station MH2 and transmitted to Sink2 as the EF traffic stream. In chapter 6, we have measured the effective bandwidth of the 5 Mbps of IEEE 802.11b wireless LAN and found the value is about 3.5 Mbps. Thus the network cannot support two CBR UDP traffic streams with 3Mbps and the congestion happens. If no network congestion control mechanism is applied, the EF traffic stream cannot get enough bandwidth.

Figure 7.8 clearly proves the effectiveness of the proposed and implemented congestion control mechanism. As soon as network congestion happens, station MH2, through which the EF traffic is forwarded, broadcasts the Data Rate Decreasing Request packet. Station MH1,

through which the BE traffic is transmitted, decreases the data rate of the HTB queuing algorithm and yields bandwidth to the EF traffic. We can clearly observe from Figure 7.8 that the bandwidth allocated to BE traffic is decreased to a small value sharply and the bandwidth allocated to EF traffic is maintained at 3Mbps after 9th second. The observed bandwidth changing is the result of the "fast decrease and slow increase" work circle of the AIMD algorithm[13]. When the bandwidth of EF traffic is reduced to 2 Mbps at the 61st second, the bandwidth of the BE traffic increases correspondingly to utilize the available bandwidth. The increase speed is determined by the values assigned to increasing factor and rate-updating interval. This is investigated in the following sections.

7.2.2 Test case 2: Assessing the Effect of Decreasing Factor on the Performance

When the measured NIC delay d_{NIC} exceeds the set threshold value d_{th}, the MH2 station broadcasts a data rate decreasing request message. The other station i, after receiving this message, reduces the data rate of the BE traffic from $R^i_{BE}(t)$ to $R^i_{BE}(t+T)$ as:

$$R^i_{BE}(t+T) = R^i_{BE}(t) \times r^i_{de} \tag{7.2}$$

where r^i_{de} is the decreasing factor on station i, which is less than 1. From equation (7.2), we can see that the smaller the decreasing factor, the larger the reduction of BE traffic becomes. Thus the bandwidth which is yielded from the BE traffic to the EF traffic is larger. In this case, the required bandwidth of EF traffic is satisfied more quickly. However, the service on EF traffic is given a priority at the expense of degradation of BE traffic. An increasing factor is used to reduce this degradation of BE traffic, which will be investigated in the next section.

In what follows, we report results collected for value of decreasing factor 0.25, 0.5 and 0.75.

[13] This algorithm is described in chapter 4.

Table 7.2 Configuration of parameters and traffic sources used in test case 2

Parameter/ traffic source names	**Value**
Decreasing factor r_{de}^{i}	0.25, 0.5, 0.75
Increasing factor C_{in}^{i} (Kbps)	50 Kbps
Desired delay d_{th} (ms)	1.2 ms
Data rate-updating interval T (ms)	200 ms
EF traffic	3 Mbps CBR UDP
BE traffic	3 Mbps CBR UDP

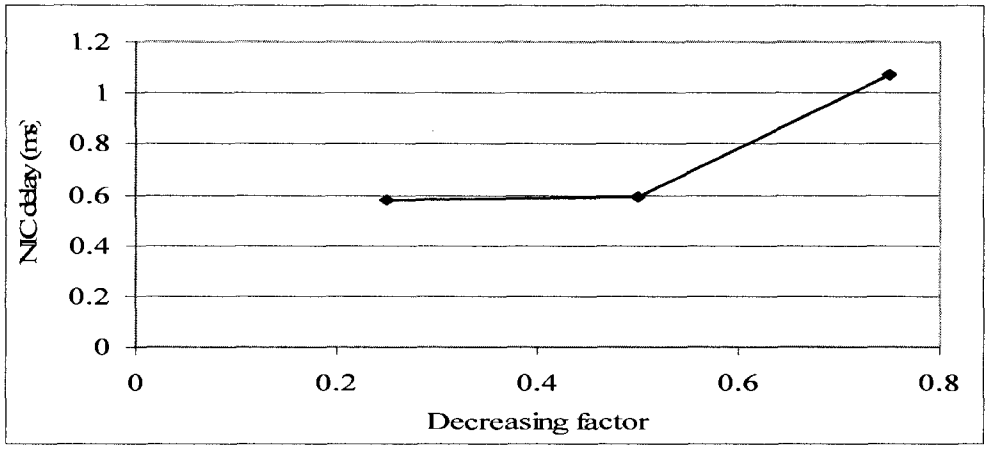


Figure 7.9 Average NIC delay as experienced by the EF traffic versus the value of decreasing factor r_{de}^{i}

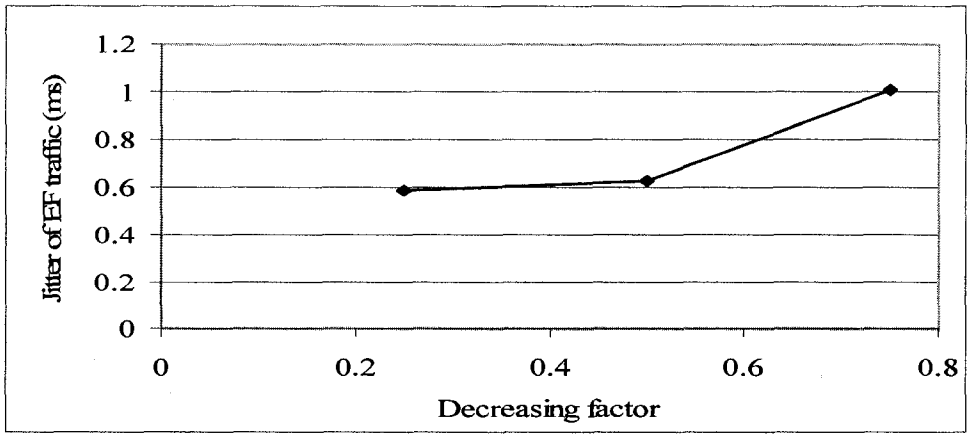


Figure 7.10 Jitter of the NIC delay as experienced by the EF traffic versus the value of decreasing factor r_{de}^{i}

Figure 7.9 and Figure 7.10 show the average NIC delay and delay jitter of the EF traffic, respectively. We can see that for the decreasing factor less than 0.5, the average NIC delay and delay jitter experienced by the EF traffic are small (average NIC delay as 0.58 ms and delay jitter as 0.6 ms).

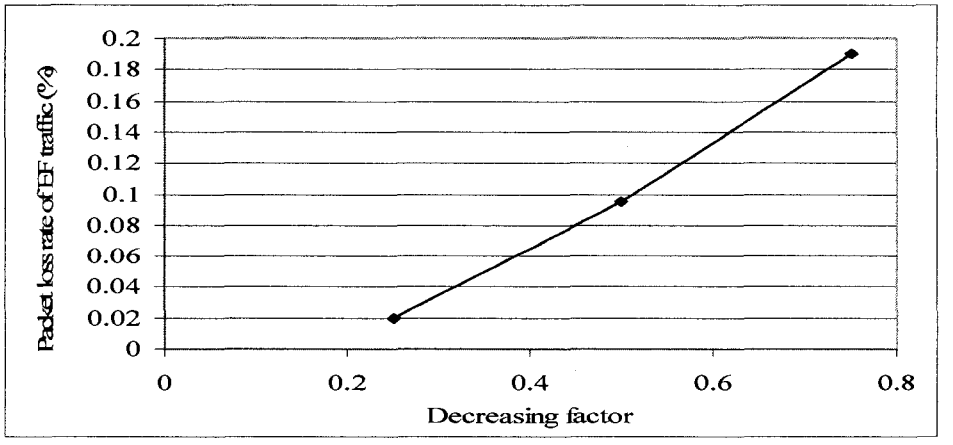


Figure 7.11 Packet loss rate as experienced by the EF traffic versus the value of decreasing factor r_{de}^{i}

Figure 7.11 shows the packet loss rate of the EF traffic versus the decreasing factor. It is evident that the system can support applications requiring loss rate 10^{-1} (1%), such as Voice Over IP. In addition, with proper choice of the decreasing factor, the system can support applications that are sensitive to packet losses, such as Video Conference (a packet loss rate in the rage of 10^{-4} is required).

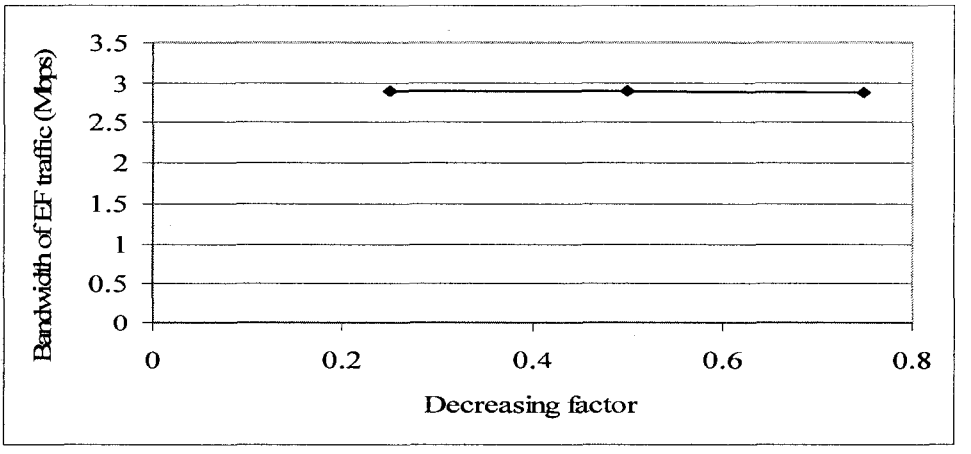


Figure 7.12 Average bandwidth consumed by the EF traffic versus the value of decreasing factor r_{de}^{i}

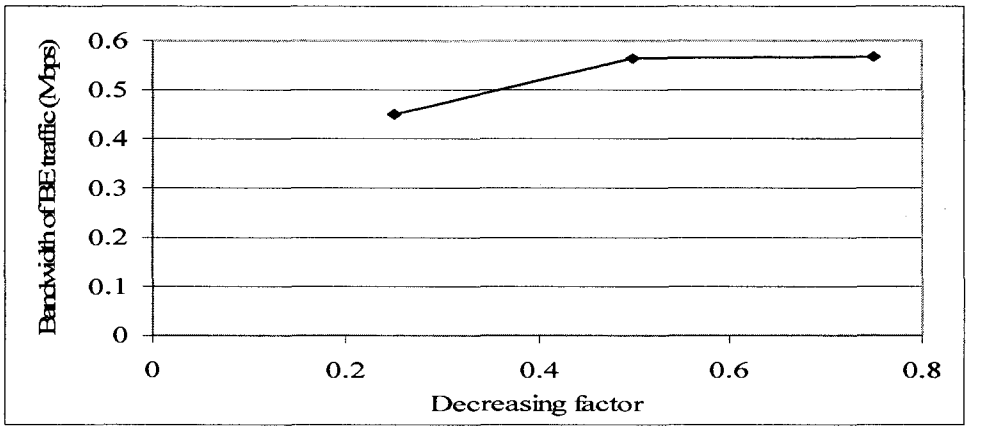


Figure 7.13 Average bandwidth consumed by the BE traffic versus the value of decreasing factor r_{de}^{i}

Figure 7.12 and Figure 7.13 show the average bandwidth consumed by EF traffic and BE traffic respectively, versus the decreasing factor. We can see that the average bandwidth of the EF traffic is almost constant close to 3 Mbps, independent of the value of the decreasing factor. On the other hand, the average bandwidth of the BE traffic increases with the decreasing factor increases.

We can conclude from above figures that the value 0.25 is a good choice for the decreasing factor. When using this value, the QoS scheme is able to guarantee the requirement of Packet loss rate and bandwidth for the EF traffic steam.

7.2.3 Test case 3: Assessing the Effect of Increasing Factors on the Performance

This section investigates the impact of the value of the increasing factor c_{in}^{i} has on the performance of the proposed QoS scheme. As mention in section 4.3, when the measured NIC delay d_{NIC} is bellow desired delay d_{th}, the data rate of the BE traffic is increased from $R_{BE}^{i}(t)$ to $R_{BE}^{i}(t+T)$, as following expression:

$$R_{BE}^{i}(t+T) = R_{BE}^{i}(t) + c_{in}^{i} \qquad (7.2)$$

Table 7.3 Configuration of parameters and traffic sources in test case 3

Parameter/ traffic source names	**Value**
Decreasing factor r_{de}^{i}	0.25
Increasing factor c_{in}^{i} (Kbps)	25 Kbps, 50 Kbps, 75 Kbps
Desired delay d_{th} (ms)	1.2 ms
Data **rate-updating** interval (ms)	200 ms
EF traffic	3 Mbps CBR UDP
BE traffic	3 Mbps CBR UDP

The parameters of the QoS scheme and the traffic source used in this test case are shown in table 7.3. We simulate three values of the increasing factor: 25 Kbps, 50 Kbps and 75 Kbps. Figure 7.14 to Figure 7.18 provide the average NIC delay, delay jitter, packet loss and average bandwidth of the EF traffic versus the different values of increasing factor, respectively.

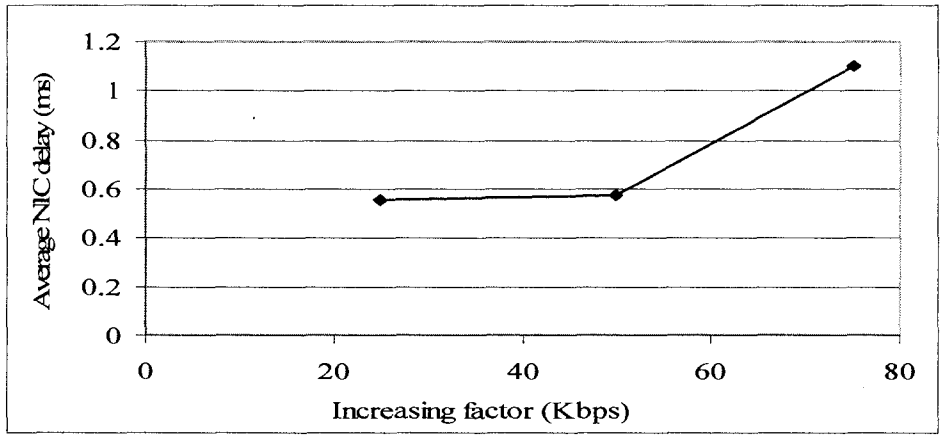


Figure 7.14 Average NIC delay as experienced by EF traffic versus the value of increasing factor c_{in}^{i}

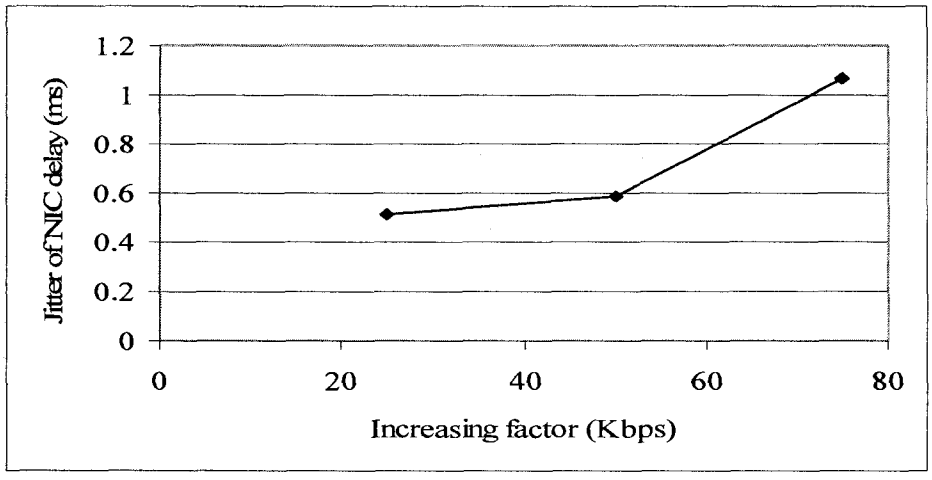


Figure 7.15 NIC delay jitter as experienced by the EF traffic versus the value of increasing factor c_{in}^{i}

Figure 7.14 and Figure 7.15 show the average NIC delay and delay jitter experienced by the EF traffic versus the increasing factor. We can see that the average NIC delay is kept small (around 0.6 ms) when the increasing factor is 25 Kbps or 50 Kbps. When the increasing factor is 75 Kbps, the average delay is increased to 1.1ms, which is still less than the desired delay (1.2 ms). The delay jitter is also a strict monotonically increasing function of the increasing factor. Therefore, with a larger increasing factor, the average delay and delay jitter of the EF traffic increase.

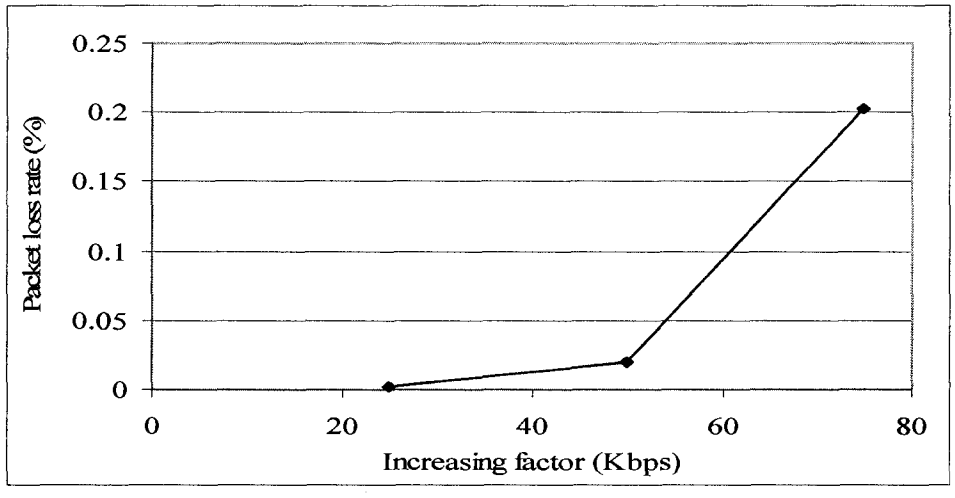


Figure 7.16 Packet loss rate of the EF traffic versus the value of increasing factor c_{in}^{i}

Figure 7.16 shows the packet loss rate experienced by the EF traffic versus the increasing factor. We can see that when the increasing factor is 25 Kbps or 50 Kbps, the packet loss rate of the EF traffic is less than or around 10^{-3}. When the increasing factor is 75 Kbps, the packet loss rate is increased to 0.2%. That means, in that case, the increased bandwidth of the BE traffic is larger than the available network resource. Therefore, the bandwidth of the EF traffic is reduced and its performance is degraded.

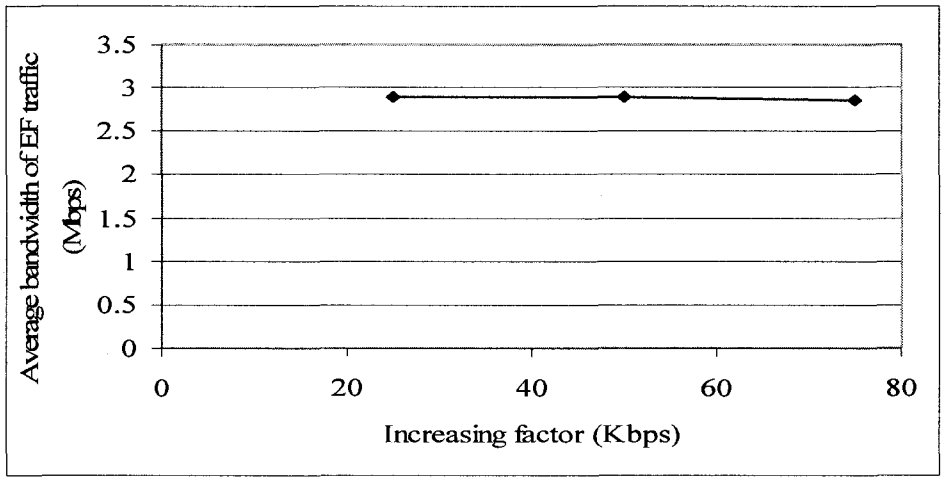


Figure 7.17 Average bandwidth consumed by EF traffic versus the value of increasing factor c_{in}^{i}

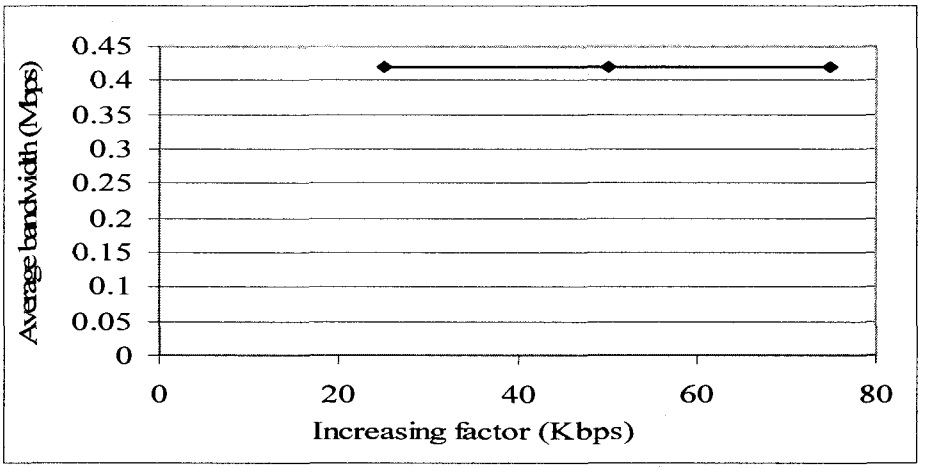


Figure 7.18 Average bandwidth consumed by the BE traffic versus the value of increasing factor c_{in}^{i}

Figure 7.17 and Figure 7.18 show the average bandwidth consumed by the EF and BE traffic respectively, versus the increasing factor. From Figure 7.17, we can see that the average bandwidth of the EF traffic remains unchanged in respect to the value of the increasing factor, its value been close to the date rate of 3 Mbps.

7.2.4 Test case 4: Assessing the Effect of Data Rate-updating Interval on the Performance

The purpose of this test case is to assess the impact of the rate-update interval on the performance of the proposed QoS scheme. As explained in Chapter 4, the data rate controller updates the data rate every T milliseconds. The way the rate should be updated depends on the values of the measured NIC delay d_{NIC} and the desired delay d_{th}. Smaller updating interval would enable the controller to react faster to the changing traffic and channel conditions. However, a smaller update interval would increase the process load imposed on the system, since the controller's algorithm would be run more frequently.

The parameters of the QoS scheme and of the traffic sources used in this test case are shown in table 7.4. We have run experiments for three values of the update interval: 100 ms, 200 ms and 400 ms.

Figures 7.19 to 7.23 present the effect of the data rate-updating interval on the QoS performance. Again, the average NIC delay, delay jitter, packet loss and average bandwidth of the EF traffic with the different rate-updating intervals are provided in Figure 7.19 to Figure 7.22, respectively.

Table 7.4 Configuration of parameters and traffic sources in test case 4

Parameter/ traffic source names	**Value**
Decreasing factor r_{de}^{i}	0.25
Increasing factor c_{in}^{i} (Kbps)	50 Kbps
Desired delay d_{th} (ms)	1.2 ms
Data rate-updating interval T(ms)	100 ms, 200 ms, 400 ms

EF traffic	3 Mbps CBR UDP
BE traffic	3 Mbps CBR UDP

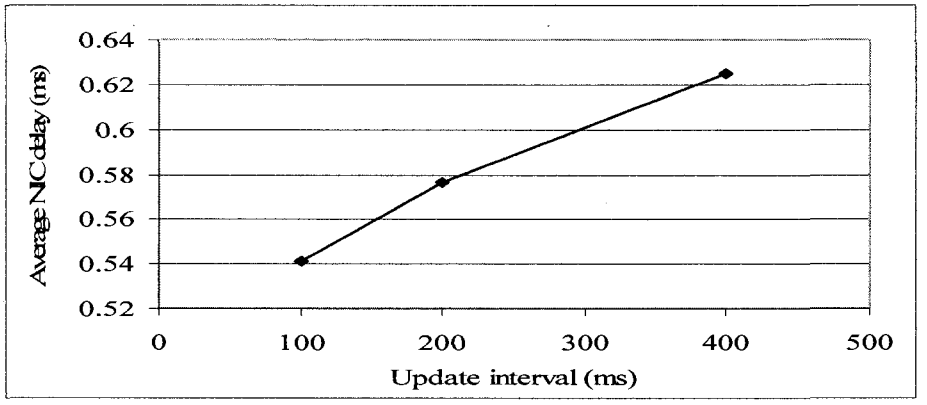


Figure 7.19 Average NIC delay as experienced by EF traffic versus the data rate-updating interval T

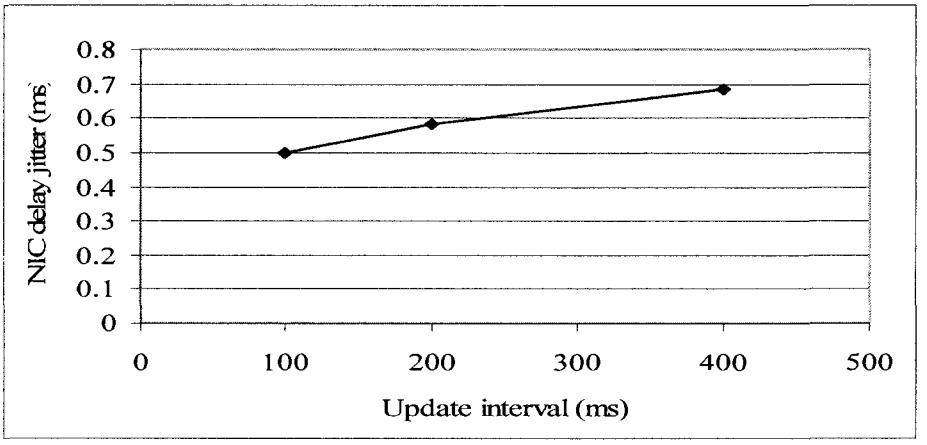


Figure 7.20 NIC delay jitter as experienced by EF traffic versus the data rate-updating interval T

Figure 7.19 and Figure 7.20 show the average NIC delay and delay jitter experienced by the EF traffic, respectively. We can see that by decreasing the rate-updating interval, the average NIC delay and delay jitter decreases. This is an expected result. As the controller observes the NIC delay more frequently, it is able to react earlier to a developing congestion when it is at its earlier stage.

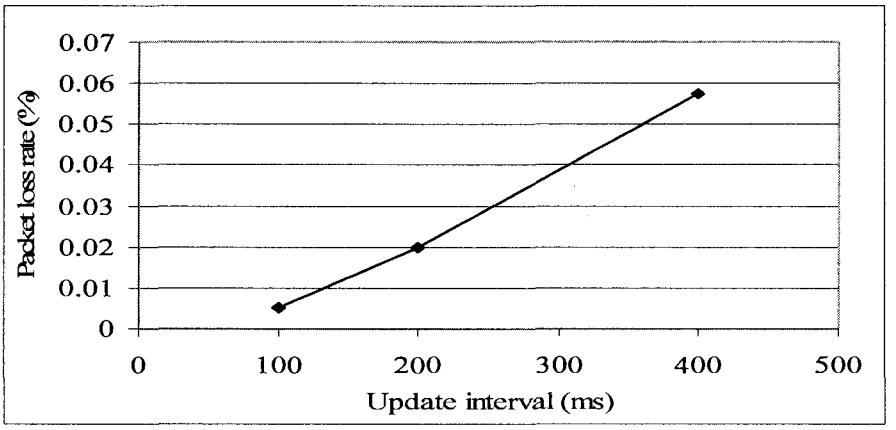


Figure 7.21 Packet loss rate of EF traffic versus the data rate-updating interval T

Figure 7.21 shows the packet loss rate of the EF traffic versus the data rate-updating interval. We can see that the packet loss rate of the EF traffic is a monotonically increasing function of the data rate-updating interval.

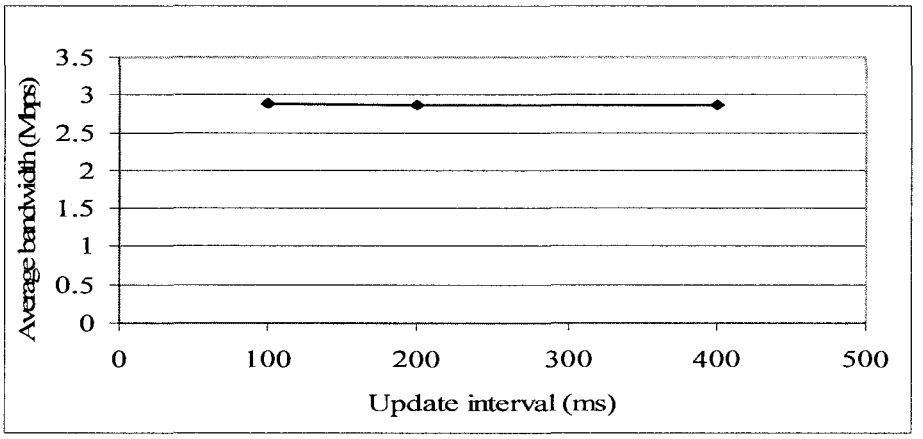


Figure 7.22Average bandwidth consumed by EF traffic versus the data rate-updating interval

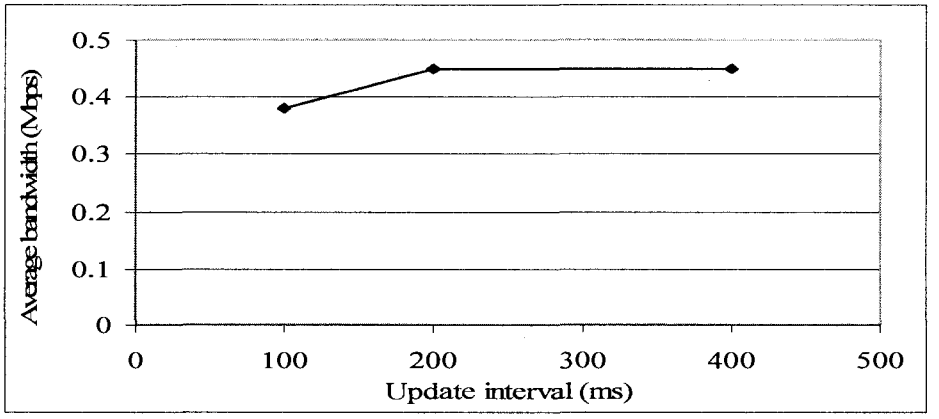


Figure 7.23 Average bandwidth consumed by BE traffic versus the data rate-updating interval T

Figure 7.22 and Figure 7.23 show the average bandwidths of the EF traffic and BE traffic versus the data rate-updating intervals, respectively. We can see that the average bandwidth of the EF traffic is almost constant and close to its transmission rate of 3 Mbps.

When the rate-updating interval is larger than 200 ms, the average bandwidth consumed by the BE traffic is constant as well, while for values lower than 200 ms.

7.2.5 Test case 5: The system response time

In the previous test cases, the performance of the distributed network congestion control mechanism is evaluated for different BE traffic values, increasing factor, decreasing factor and date rate updating interval. However, we still have one problem that has to be answered: how fast can this QoS mechanism react to the changing condition of the network congestion status?

In order to answer this question, the system response time is measured in this test case. The system response time reflects how fast the QoS scheme is able to adapt the changing of the network traffic load and provides QoS guarantee to the EF traffic. The parameters of the QoS scheme and the traffic source used in this test case are shown in table 7.5.

Table 7.5 Configuration of parameters and traffic sources in test case 5

Parameter/ traffic source names	**Value**
Decreasing factor r_{de}^{i}	0.25
Increasing factor c_{in}^{i} (Kbps)	50 Kbps
Desired delay d_{th} (ms)	1.2 ms
Data rate-updating interval T(ms)	100 ms, 200 ms, 400 ms
EF traffic	3 Mbps CBR UDP
BE traffic	3 Mbps CBR UDP
Network buffer size	100 packets

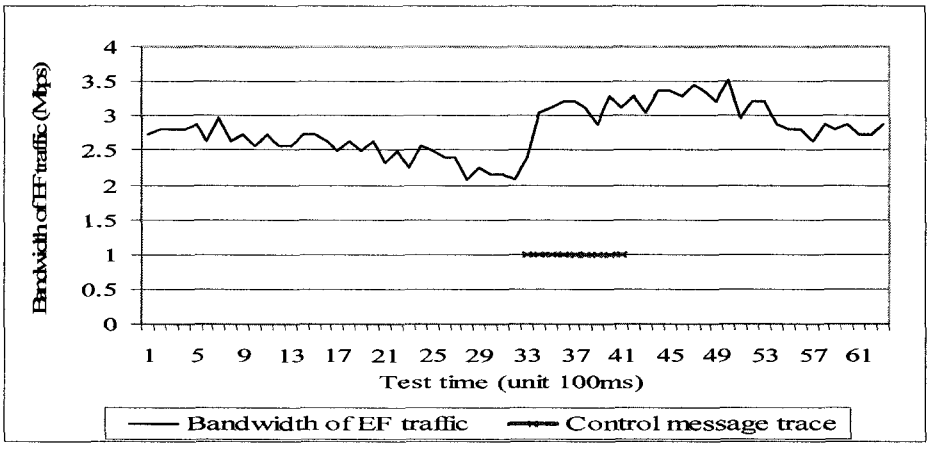


Figure 7.24 The system response time versus 10ms control message sending interval

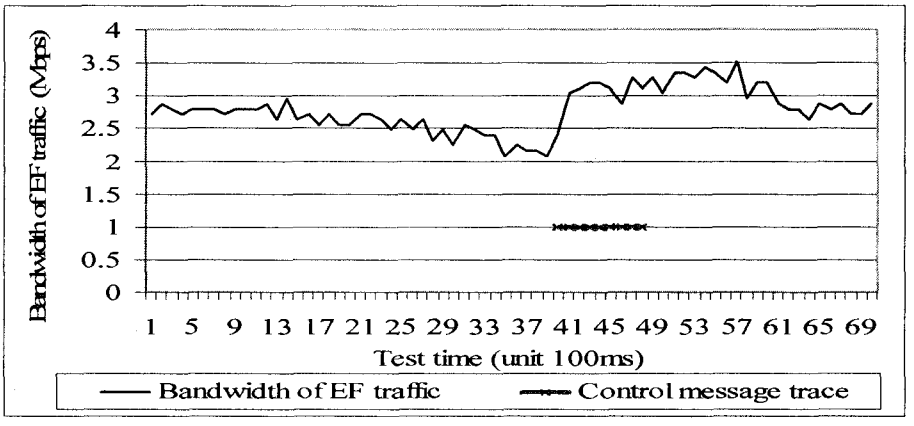


Figure 7.25 The system response time with 20ms control message sending interval

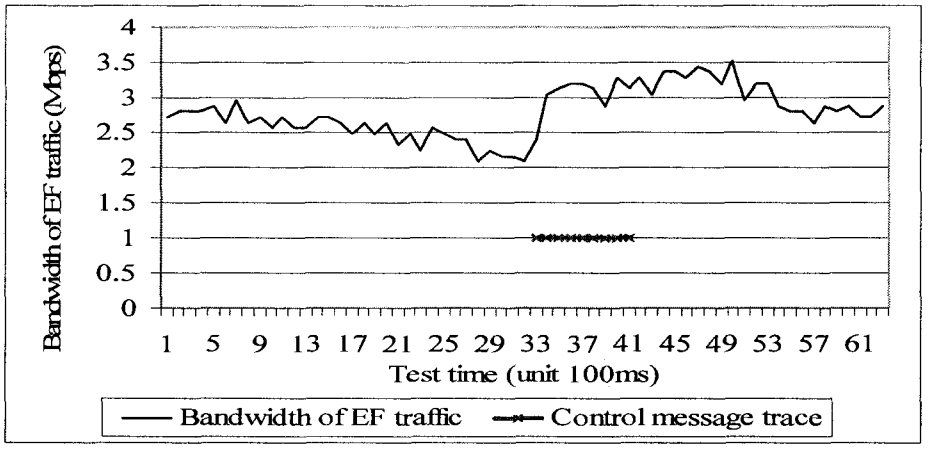


Figure 7.26 The system response time with 50ms control message sending interval

The above figures present the transient behavior of the control message and the changing of the EF traffic bandwidth while the data rate update interval is 10 ms, 20 ms and 50 ms. Please note that we include the trace of the control message in these figures in order to show the relationship between the broadcast of control message form station MH2 and the bandwidth of EF traffic transmitted on the station MH2.

From these figures, we can observe that the system response time is about 100 ms. During this response period, the following tasks are accomplished by the distributed network congestion control mechanism:

1. Station MH1 receives the Date Rate Decreasing Request Message.
2. The QoS scheme recalculates the new data rate for BE traffic (The algorithm is described in chapter 4)
3. The HTB queuing adjusts the data rate and reduce the data rate of BE traffic stream.

In the design of the distributed network congestion control mechanism, several methods have been implemented in order to reduce the response time, such as priority queuing for control message, small control message updating interval value, in_kernel HTB data rate adjusting. The performance latency will be further reduced if the real MAC delay value is used (The firmware of wireless LAN card should be modified to export the value out) and the Netlink layer beacon frame is used instead of the IP packet as the control message.

In order to investigate the delay of response time for out proposed QoS scheme, a test is designed to find out how fast the HTB queue can handle the data rate change. This test is performed on a single PC on which the HTB queuing module runs on the wireless NIC. The experiment configuration is shown in Figure 7.27.

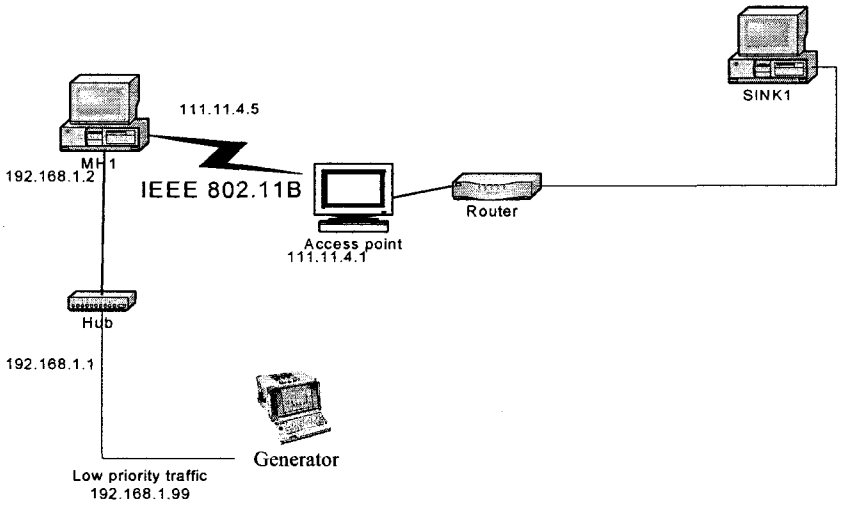


Figure 7.27 Network configuration in HTB response time test

The network generator starts UDP CBR data traffic at the speed of 2.8 Mbps. The traffic is forwarded at MH1. The entry NIC is eth1, which is configured as 192.168.1.2. The packet is forward the wireless NIC, wlan0, which is configured as 192.168.4.5 The HTB queuing module is enabled at wlan0. At the time 0, we set the data rate of HTB queue as 3 Mbps. Then at time 1400th ms, we use the tc command to set the rate to 0.5 Mbps. The change curve of the UDP traffic date rate is recorded and shown in the Figure 7.26.

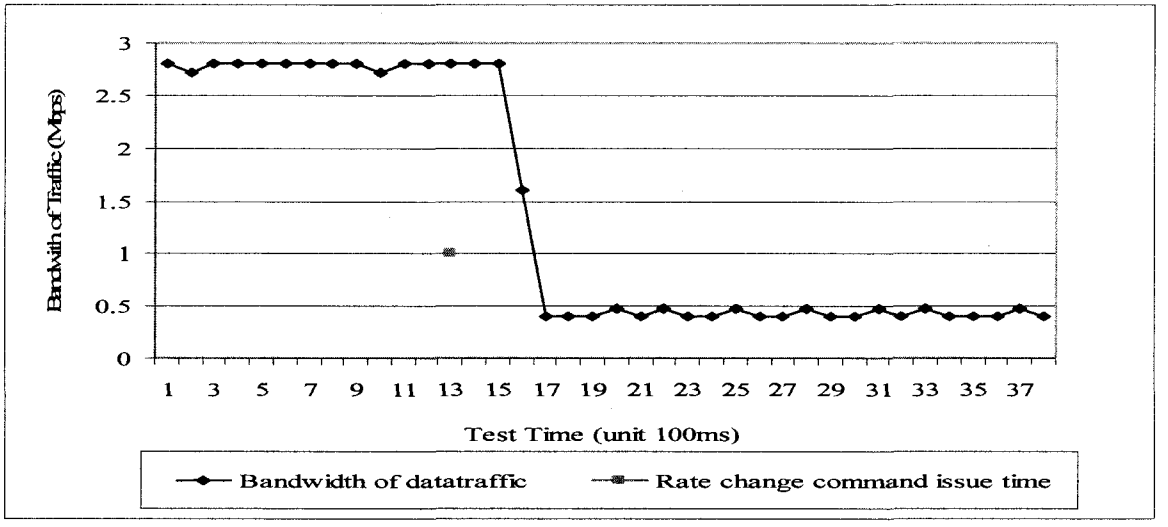


Figure 7.28 The response time for changing of HTB queue rate

From Figure 7.28, we can observe that after the tc command is issued at 1400th ms, the data rate of the UDP traffic was decreased from 2.8 Mbps to 0.5 Mbps until 1700th ms. This result shows that there is latency existed in the HTB queue and it is one cause of the response delay in our QoS scheme.

7.3 Summary

In this chapter, we presented the simulation results of four test cases, in order to show the power of our QoS schedule. Furthermore, we investigated the effects of its three parameters, such as decreasing factor, increasing factor and data rate-updating interval, on the performance. From simulation results, we can see that with our QoS schedule, the bandwidth requirement of the EF traffic is satisfied with priority and the bandwidth of the BE traffic is as large as possible when the network resource is available. Therefore, our QoS schedule works very well.

For each test case, we show the average NIC delay, delay jitter, packet loss rate and average bandwidth of the EF traffic and the average bandwidth of the BE traffic as well. The performance of our QoS schedule is mainly determined by three parameters: decreasing factor, increasing factor and data rate-updating intervals. With lower decreasing factors and lower increasing factors, the bandwidth of the EF traffic is guaranteed much better. However, the network resource may not be used efficiently. With a larger increasing factor, the available network resource can be assigned to the BE traffic. Small data rate-updating intervals make sure that our QoS schedule responds to the network condition in time.

Chapter 8

Conclusion

This thesis investigated the performance of IEEE 802.11b wireless LAN and proposed a QoS capable wireless LAN architecture, which is able to provide QoS service for the multimedia applications deployed on the wireless LAN.

The capacity of IEEE 802.11b wireless LAN is evaluated in an indoor environment. The test cases include UDP traffic performance on peer-peer and multi-stations configuration. In order to decide the influence of dynamic changing wireless channel condition on the performance of IEEE802.11, we also run the test case on a roaming mobile station and in an environment full of radio noise. The test results showed that the signal obstacles (such as wall) and radio noise had big interference to the performance of IEEE802.11 wireless LAN. For example, if one of the mobile stations works in a bad wireless environment, the retransmission of it's damaged frames will consume a lot bandwidth and downgrade the performance of whole wireless LAN.

We developed a low cost/complexity QoS capable wireless LAN architecture. The proposed distributed network congestion control scheme is based on the local network interface (NIC) packet transmission delay. This QoS scheme is developed on top of the IEEE802.11 DCF function, which means that it can be easily implemented using commercially available wireless LAN products. It only involved new software of the Linux IP network stack, not the link layer. Thus the wireless LAN card does not need to be modified.

The prototype of the proposed technique was implemented on a Red Hat Linux based IEEE802.11b wireless LAN testbed. The Orinoco network card driver and Linux kernel code are modified to realize the features of our QoS scheme. Its performance was assessed

through carefully designed test cases. Thorough experiments were also completed to assess the system sensitivity to the values of several key parameters.

The prototype evaluation test result shows that the QoS capable architecture can provide effectible QoS service for the EF traffic based on standard IEEE 802.11 network card.

Future work

1. The QoS capable architecture prototype will be further developed to make it easy to configuration. Also the testbed will be extended to include more handheld devices such as iPaqs.
2. The capacity performance test will be done on the IEEE802.11a/g wireless LAN to investigate how the QoS scheme works on the two new wireless LAN standards.
3. Implemented the proposed QoS mechanism in layer 2 (MAC layer, realized in network hardware) instead of in layer 3 (Network layer, realized in network card driver and IP packet).

Reference

[ABH93] A. Parekh and R. Gallage, "A Generalized Processor Sharing Approach To Flow Control in Integrated Services Networks: The Single-Node Case", *IEEE/ACM Transactions on Networking*, 1(3), Jun. 1993.

[AGE00] Agere Systems. User's guide for Orinoco pc card, Sept. 2000. ftp://ftp.orinocowireless.com/pub/docs/ORINOCO/MANUALS/ug_pc.pdf.

[AHN02A] Gahng-Seop Ahn, A. Campbell, A. Veres, Li-Hsiang Sun, "Supporting Service Differentiation for Real-Time and Best-Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN)", *IEEE Trans. on Mobile Computing*, vol. 1, no. 3, July -- September 2002.

[AHN02B] Gahng-Seop Ahn, A. Campbell, A. Veres, and Li-Hsiang Sun, "SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks," Internet Draft, <draft-ahn-swan-0.1.txt>, work-in-progress, Sept.2002.

[ALM99] W. Almesberger, "Linux Network Traffic Control—Implementation Overview", White Paper, April 1999.

[ALM99] W. Almesberger, J. Salim, and A. Kuznetsov. "Differentiated Services on Linux", June 1999. http://diffserv.sourceforge.net.

[BAN02] A. Banchs, X. Perez, "Distributed Weighted Fair Queuing in 802.11 Wireless LAN", Proc. *IEEE ICC*, Apr. 2002.

[BIA00A] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function", *IEEE J. Selected Areas in Comm.*, vol. 18, Mar. 2000.

[BIA00B] G. Bianchi, A. Capone, and C. Petrioli, "Throughput Analysis of End-to-End Measurement-Based Admission Control in IP", *Proc. IEEE INFOCOM '00*, 2000.

[BLA98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "Rfc2475 – An Architecture For Differentiated Services", Informational Category, Dec. 1998.

[BRA94] R. Braden, D. Clark, and S. Shenker, "Rfc1633 - Integrated Services in the Internet Architecture: An Overview", Informal Category, Jul. 1994.

[BRA97] R. Braden, L. Zhang, S. Bersion, S. Herzog, and S. Jamin, "Rfc2205 – Resource Reservation Protocol - Version 1 Functional Specification", Standards Track, Sept. 1997.

[BRE00] L. Breslau, E. Knightly, S. Shenker, I. Stoica, and H. Zhang, "Endpoint Admission Control: Architectural Issues and Performance", *Proc. ACM SIGCOMM '00*, Aug. 2000.

[CAL98] F. Calì, M. Conti, E. Gregori, "IEEE 802.11 Wireless LAN: Capacity Analysis and protocol Enhancement", *Proc. of INFORCOM 1998*, pp 142-149.

[CAO01] Y. Cao, V. Li, "Scheduling Algorithms in Broad-Band Wireless Networks", *IEEE Proceedings of the IEEE*, VOL.89, No1, January 2001.

[CEN00] C. Centinkaya and E. Knightly, "Egress Admission Control", *Proc. IEEE INFOCOM '00*, 2000.

[CHH97] H. Chhaya, S. Gupta, "Performance Modeling of Asynchronous Data Transfer Methods in the IEEE 802.11 MAC protocol", *ACM/Balzer Wireless Networks,* vol. 3, 1997, pp. 217-234.

[CRO97] B. Crow, I. Widjaja, J. Kim, P. Sakai, "IEEE 802.11 Wireless Local Area Networks: Capacity Analysis", *IEEE Communications Magazine*, September 1997.

[DEM89] A. Demers, S. Keshav and S. Shenker, "Analysis And Simulation of a Fair Queuing Agorithm", *Proceedings of SIGCOMM'89*, pp. 3 – 12, 1989.

[ECK96] D. Eckhardt, P. Steenkiste, "Measurement and Analysis of The Error Characteristic of An In-building Wireless Network", *Proceedings of SIGCOMM'96.*

[ECK00] D. Eckhardt and P. Steenkiste, "Effort-limited Fair (ELF) Scheduling for Wireless Networks", *IEEE INFOCOM 2000.*

[EUG98] T. Eugene, I. Stoica and H. Zhang, "Packet Air Queuing Algorithms for Wireless Networks With Location-Dependent Errors", *Proceedings of IEEE INFOCOMM*, pp. 1103 -- 1111, 1998.

[FANG02] Z. Fang, B. Bensaou, and Y. Wang, "Performance evaluation of a fair backoff algorithm for IEEE 802.11 DFWMAC", *Mobile Computing and Networking*, pages 48–57, 2002.

[FAT02] H. Fattah, C. Leung, "An Overview of Scheduling Algorithm in Wireless Multimedia Networks", *IEEE Wireless Communication*, October 2002.

[HAD02] Z. Velkov and B. Spasenovski, "Capture Effect in IEEE 802.11 Basic Service Area Under Influence of Rayleigh Fading and Near/Far Effect", *The 13th IEEE International symposium on Personal, Indoor and Mobile Communications*, 2002.

[HEI99] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Rfc2597 - Assured Forwarding PHB Group", Standards Track, Jun. 1999.

[HUB01] B. Hubert, G. Maxwell, R. Mook, M. Oosterhout, P. Schroeder, and J. Spaans, "Linux 2.4 Advanced Routing Howto", Apr 2001. http://www.ds9a.nl/2.4Routing.

[GOL94] S. J. Golestani, "A Self-Clocked Fair Queuing Scheme For Broadband Applications", *IEEE INFOCOM,* 1994.

[IW95000] Interwatch 95000 network analyzer/generator http://www.nettest.com/

[JAC99] V. Jacobson, K. Nichols, and K. Poduri, "Rfc2598 - an Expedited Forwarding PHB", Standards Track, Jun. 1999.

[JER00] M. C. Jeruchim, P. Balaban and K. S. Shanmugan, "Simulation of Communication Systems, second edition", Kluwer Academic Publishers, 2000.

[KEL00] F. Kelly, P. Key, and S. Zachary, "Distributed Admission Control", *IEEE J. Selected Areas in Comm.*, vol. 18, no. 12, Dec. 2000.

[KES91] S. Keshav, "On the Efficient Implementation of Fair Queuing", *Journal of Internetworking Research and Experience*, 2(3), 1991.

[KUZ] A. Kuznetsov, "Iproute2 Package". ftp://ftp.inr.ac.ru/ip-routing/.

[LEO00] A. Garcia, I. Widjaja, "Communication Networks Fundamental Concepts and Key Architectures", McGraw_Hill,2000.

[LIAO01] R. Liao, "A Utility-Based Approach for Quantitave Adaption in Wireless Packet Networks", *Wireless Networks 7, 2001.*

[LU97] S. Lu, V. Bharghavan and R. Srikant, "Fair Scheduling in Wireless Packet Networks", *Proceedings of ACM SIGCOMM'97*, 1997.

[LU99] S. Lu, T. Nandagopal, and V. Bharghavan, "Design and Analysis of An Algorithm for Fair Service In Error-prone Wireless Channels", *Wireless Networks Journal*, Feb. 1999.

[MANG02] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, L. Stibor, "IEEE 802.11e Wireless LAN for Quality of Service", *Proceedings of the European Wireless*, volume 1, pages 32–39, Florence, Italy, February 2002.

[MANN02] J. Manner, "Evalution of Mobility and Quality of Service Interaction", *Computer Networks 38 (2002).*

[MAR02] M. Devik, "HTB for Linux", http://luxik.cdi.cz/~devik/qos/htb.

[MOO00] J. Moorman, J. Lockwood and S. Kang, "Wireless Quality of Service Using Multiclass Priority Fair Queuing", *IEEE Journal on Selected Areas in Communications*, Aug. 2000.

[NAN99] T. Nandagopal, S. Lu and V. Bharghavan, "A Unified Architecture For The Design and Evaluation of Wireless Fair Queuing Algorithms", *MobiCom'99 – Proceedings of The*

Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 132 – 142, Aug. 1999.

[NBL97] B. Nble, M. Satyanarayanan, "Trace-Based Mobile Network Emulation", Proceedings of the ACM SIGCOMM '97 conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, 1997

[FLO95] S. Floyd, V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks", *IEEE/ACM Trasactions on Networking*, Vol3 No.4 August 1995.

[OLS] D. Olshefski, "*TC API Beta 1.0.* IBM T.J. Watson Research", http://oss.software.ibm.com/developerworks/projects/tcapi/.

[PAT02] W. Pattara-atikom, S. Banerjee and P. Krishnamurthy, "Starvation Prevention and Quality of Service in Wireless LANs", *Proc. IEEE WPMC*, oct. 2002.

[PCMCIA] Pcmcia card services for linux. http://pcmcia-cs.sourceforge.net.

[PCAP] Packet Capture library http://www.tcpdump.org/pcap3_man.html

[PRA01] N. Prasad, A. Prasad, "WLAN Systems and Wireless IP For Next Generation Communications", ISBN 1-58053-290-X, 2001.

[RAN99] R. Ranasinghe, D. Everitt, L. Andrew, "Fair Queueing Scheduler For IEEE 802.11 Based Wireless Multimedia Networks", Multiaccess, Mobility and Teletraffic in Wireless Communications (MMT'99), Oct. 1999, vol. 4, pp. 241–251.

[RUS] R. Russell, "Linux 2.4 Packet Filtering Howto", http://netfilter.samba.org.

[RTN] Linux manpage, RTNetlink (7) http://www.squarebox.co.uk/cgi-squarebox/manServer/usr/share/man/man7/rtnetlink.7]

[RUB01] A. Rubini, J. Corbet, "Linux Device Drivers, 2nd Edition", June 2001 0-59600-008-1

[SHR95] M. Shreedhar, G. Varghese, "Efficient Fair Queuing Using Deficit Round Robin", *Proceedings of SIGCOMM'95*, pp. 231 – 242, 1995.

[SOB99] J. Sobrinho and A. Krishnakumar, "Quality-of-Service in Ad Hoc Carrier Sense Multiple Access Networks", *IEEE J. Selected Areas in Comm.*, vol. 17, no. 8, pp. 1353-1368, Aug. 1999.

[STE94] W. Richard Stevens, "TCP/IP Illustrated: The Protocols", Addison-Wesley Publishing Company, 1994. ISBN 0-201-63346-9 (v.1).

[STE98] W. Stevens, "Unix Network Programming network APIs: sockets and XTI", Prentice HALL, Inc, 1998

[STO97] I. Stoica, H. Zhang, and T. Eugene Ng, "A Hierarchical Fair Service Curve Algorithm for Link-sharing, Real-time and Priority Service", *Proceedings of SIGCOMM'97*, 1997.

[TOU] J. Tourrilhes, "Linux Wireless Extention/Wirleess Tools", http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/.

[VAI00] N. Vaidya, P. Bahl, and S. Gupta, "Distributed Fair Scheduling In A Wireless LAN", Mobile *Computing and Networking*, pages 167–178, 2000.

[VER01] A. Veres, A. Campbell, M. Barry, "Supporting Service Differentiation in Wireless Packet Networks Using Distributed Control", *IEEE J. Selected Areas in Comm*, vol. 19, no. 10, OCTOBER 2001.

[WIS01] L. Wischhof, J. Lockwood, "Packet Scheduling for Link-Sharing and Quality of Service Support in Wireless Local Area Networks", Technical Report, WUCS 01-35, Washington University, 2001

[WLANA] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, High-speed Physical Layer in the 5 GHz Band, IEEE 802.11b-1999. (Supplement to ANSI/IEEE, std 802.11, 1999 edition).

[WLAN] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, ANSI/IEEE, Std 802.11, 1999 edition.

[WLANB]Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band, IEEE 802.11b-1999. (Supplement to ANSI/IEEE, std 802.11, 1999 edition.)

[WU02] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma, "Performance of Reliable Transport Protocol over IEEE 802.11 Wireless LAN: Analysis and Enhancement", Proceedings *of the IEEE INFOCOM'02*, June 2002.

[XYL99] G. Xylomenos and G. Polyzos, "TCP and UDP Performance Over a Wireless LAN", Proceedings *of the IEEE INFOCOM'99*, pages 439–446, March 1999.

[YU01] H. Yu, D. Makrakis, L. Orozco-Barbosa, "Experimental Evaluation of MPE-2 Video over Differentiated Services IP Networks", *Proceedings of Pacific Rim 2001 Conference*, Victoria, Aug. 2001.

[ZHA01] Y.Zhang, "Network Architecture For Dynamic Support of Service Differentiation in 4th Generation Broadband Wireless Networks", University of Western Ontario, 2001.